BP-GAN: Bibliophile GANs for Generating Book Covers

Ankit Dwivedi, Arunothia Marappan
Stanford University
ankitd@stanford.edu, arunothi@stanford.edu

Abstract

In this project, we explore Generative Adversarial Networks (GANs) to synthetically generate images of book covers. Specifically, we experiment with a combination of different GAN variants, architectures, loss functions, and other techniques on a dataset consisting of cover images from 6 book genres. We evaluate these combinations through visual inspection of the quality of generated images. We also measure their performance using the Fréchet Inception Distance. We conclude that more work is needed to generate realistic book cover designs, but our results are promising enough to induce further research in this novel application of generative models.

1. Introduction

"Don't judge a book by its cover" is a popular English idiom. But the reality is that human beings are visual creatures. We can be immediately reached, engaged and moved by color and shape, because images enter our brain literally at light speed. As a consequence, book covers play a crucial role in connecting with the target audience. In order to establish that connection, it is common for authors and designers to take inspiration from the most popular books in the concerned genre [1] ,[2]. This is probably why we see some visual styles, themes and motifs recurring in books belonging to a specific genre.

In our work, we investigate if deep generative models can learn these recurring styles and patterns, and generate new designs for book covers. Concretely, we experiment with a few of the many variants of Generative Adversarial Networks (GANs) [13] for this task. The advent of GANs has led to a flurry of research in the field of image generation, which has resulted in state-of-the-art image generation models on a range of datasets. Our work attempts to study if some of these models can be reused, fine-tuned, or redesigned for book covers.

Besides the genre, a lot of other factors go into the making of book covers, such as the plot, characters, local and regional styles and preferences etc. We believe that the ultimate goal of this project and any potential future research in this space should not be a model that generates ready-to-use book covers and provides an alternative to human designers. Rather, we hope this project is a small step in the direction of helping designers take inspiration from not just the prevalent styles and themes in a particular genre, but also from a large pool of machine generated sample designs. The fusion of all these different sources entails the possibility of fresh, richer, diverse and even more aesthetic book covers.

2. Related Work

Since the breakthrough of deep convolutional networks in computer vision, especially on the task of image classification, there has been some research done to apply them on images of book covers for interesting book related applications such as determining the genre (Iwana et al. [16], Buczkowski et al. [11]), predicting the book's likability (Maharjan et al. [18]) etc. But the field of generating new book cover images using deep learning is still a relatively unexplored application.

GANs were introduced by Goodfellow et. al [13]. Since then, many variations have been proposed. These have helped in increasing the training stability (WGAN [9], WGAN-GP [14]), improving quality of generated images (DCGAN [23]), generating images conditioned on class labels (CGAN [20], AC-GAN [22], FC-GAN [17]), generating images from textual descriptions (GAN-CLS [24], StackGAN [27], WGAN-CLS [10]) etc.

GANs have shown significant promise in synthetically generating natural images using the MNIST, CIFAR-10, and LFW datasets. However, as Tan et al. noted in [26], all these datasets have some common characteristics: i) Most of the background/foreground are clearly distinguishable; ii) Most of the images contain only one object per image and finally iii) Most of the objects have fairly structured shapes such as numeric, vehicles, birds, face etc. Tan et al. [26] show that GANs can generate promising images even when trained on a much more challenging Wikiart dataset that consists of artistic paintings.

Genre	Train	Val	Test
Children's Books	9725	1116	2764
Computers and Technology	5703	655	1621
Cookbooks, Food and Wine	6292	722	1788
Fiction	9562	1098	2718
Science and Math	6631	761	1884
Travel	13109	1504	3725

Table 1. Preprocessed data distribution

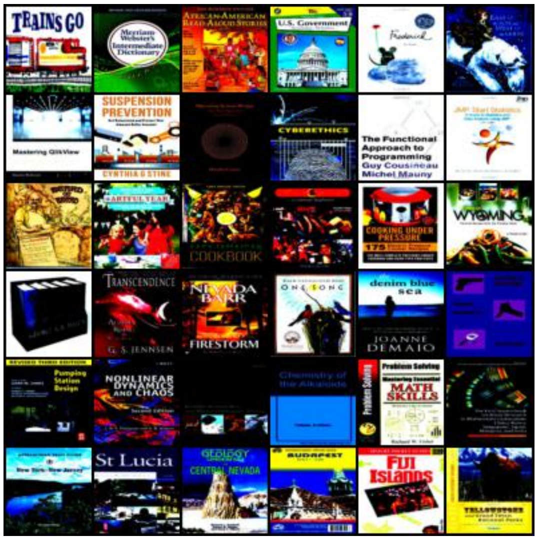

Figure 1. Input Images (64x64). Each row represents one genre in the order - Children's Books; Computers and Technology; Cookbooks, Food and Wine; Fiction; Science and Math; Travel.

3. Dataset

For training and evaluation, we use the Book Cover Dataset prepared by Iwana et al. [16]. It has more than 207K images across 32 genres. Of these, we selected 6 genres with most recognizable styles and patterns across books. The final dataset has the distribution of genres as shown in Table-1 and Figure-1. As the images were of varying shapes and sizes, we added a preprocessing step to resize them to 64 x 64 images. All our models have generated 64 x 64 images. Additionally, we normalize the image pixels to values between -1 and 1.

Similar to the Wikiart dataset used by Tan et al. [26], this is a challenging dataset for GANs, as the images in each genre are quite vivid and diverse, and don't have structured shapes or easily recognizable patterns.

4. Methods

We start this section with a brief introduction to GANs, followed by important technical highlights of the GAN variants that we experimented with for our problem. We also call out the architecture/design changes and other techniques that we employ to some of these variants. All our code is available at this Github repository - [3]. The different code repositories we referred for implementing and evaluating the models have been called out in the Acknowledgements section.

4.1. Generative Adversarial Networks

A typical generative adversarial network [13] consists of a generator network *G* and a discriminator network *D*. They act as competitors. Given training data x, *G* takes input from a random noise z and tries to generate data that has similar distribution as x. The network *D* takes input from both training data x and generated data from *G*, and estimates the probability of a sample coming from training data or generated by *G* (real or fake).

G and *D* are trained simultaneously to maximize their respective objectives. In other words, *D* and *G* play the following two-player minimax game with value function *V(G, D)*:

$$\min_G \max_D V(G, D) =$$

$$\mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log 1 - D(G(z))] \quad (1)$$

4.2. Baseline - DCGAN

DCGAN or Deep Convolutional GAN [23] was the first GAN variant that employed a deep convolutional architecture in both the generator and discriminator networks. In particular, the generator network consists of 4 blocks of ConvTranspose2D - BatchNorm2D - ReLU, followed by a final block of ConvTranspose2D and tanh activation. The discriminator has a symmetric downsampling network, but uses LeakyReLU instead of ReLU, and a final sigmoid layer for classification. The architecture is illustrated in Appendix A.

For our baseline, we trained a DCGAN model for each genre separately, and one for all genres combined. Each model was trained for 500 epochs.

4.3. Conditional GAN

Mirza et al. [20] proposed an extension to generative adversarial nets, making them conditional models by conditioning both the generator and discriminator on some extra information y. y could be any kind of auxiliary information, in our case y represents label of a specific book genre.

In the generator the prior input noise $p_z(z)$, and y are combined in joint hidden representation. In the discriminator x and y are presented as inputs. So the objective function of the two-player minimax game presented in Eq.1 changes to:

$$\min_G \max_D V(G, D) =$$

$$\mathbb{E}_{x \sim p_{data}(x)}[\log D(x|y)] + \mathbb{E}_{z \sim p_z(z)}[\log 1 - D(G(z|y))] \quad (2)$$

4.4. Conditional DCGAN

The regular DCGAN model presented in section 4.2 can be easily made conditional with the changes proposed in section 4.3:

1. Concatenate the noise vector with the one hot vector representation of the class label before feeding it into the generator.

2. Reshape the one hot label vector to $C \times H \times W$, where H and W are the image spatial dimenstions and C is the number of image channels. Both the image and the reshaped label vectors are fed into the discriminator. The first 2D convolutional layer of the regular DCGAN is split into 2 layers, each with half the number of original kernels. The first conv layer takes the image as input, and the second takes the label as input. The resulting activation maps from these 2 layers are then concatenated along the channel dimension. The rest of the network remains the same as the regular DCGAN.

4.5. Auxiliary Classifier GANs (AC-GAN)

Odena et al. [22] propose a variant of the GAN architecture, in which every generated sample has a corresponding class label, $c \sim p_c$ in addition to the noise z. The generator G uses both to generate images $X_{fake} = G(c, z)$, similar to earlier conditional models. The difference is that the discriminator gives both a probability distribution over sources (real or fake) and a probability distribution over the class labels, $P(S|X), P(C|X) = D(X)$. The loss function has two parts: the log likelihood of the correct source, L_S, and the log-likelihood of the correct class, L_C.

$$L_S = E[logP(S = real|X_{real})] + E[logP(S = fake|X_{fake})] \quad (3)$$

$$L_C = E[logP(C = c|X_{real})] + E[logP(C = c|X_{fake})] \quad (4)$$

Appendix B contrasts the designs of vanilla conditional GAN and AC-GAN.

We experimented with the architecture and design of the basic AC-GAN model, and tried some techniques to stabilize the GAN training and improve quality of generated images. Next few sections talk about some of these changes:

4.5.1 Architecture

Odena et al. [22] demonstrate that high resolution samples provide class information not present in low resolution samples, and their models generate 128 × 128 images. However, in order to speed up training time to allow for quicker experimentation, we decided to generate 64 × 64 images. This required tweaking the architectures of the generator and discriminator networks. Appendix B provides details of our model's architectural hyperparameters.

4.5.2 One-sided label smoothing & Noisy labels

The idea of one-sided label smoothing is to replace the target for the real examples with a value slightly less than one. Goodfellow et al. [12] show that this prevents extreme extrapolation behavior in the discriminator, and encourages the discriminator to estimate soft probabilities rather than to extrapolate to extremely confident classification. Instead of using a fixed value for label smoothing, such as 0.9, we observed slightly better performance by using a random number, as suggested by Chintala et al. [25]. In our models, we use random numbers between 0.75 and 1.0 to smooth the real label.

Another trick suggested in [25] is to make the labels noisy for the discriminator by occasionally flipping the labels when training the discriminator. We do this with probability=0.05.

4.5.3 LeakyReLU in Generator

As highlighted by Goodfellow et al. [12], the minimax loss function of the GAN game suffers from the problem of non-convergence. This problem becomes more accute if the gradients flowing back to the generator become sparse. To mitigate this issue to some extent, Chintala et al. [25] suggest to use LeakyReLU instead of ReLU activation in both generator and discriminator networks. We followed this tip and replaced ReLU in the base AC-GAN generator with LeakyReLU.

4.5.4 Least Squares (LS) Loss Function

Mao et al. [19] found that the sigmoid cross entropy loss function in regular GANs may lead to vanishing gradients for the generator during the learning process. To overcome this problem, they propose the least squares loss functions for the generator G and discriminator D, given by the equations:

$$\ell_G = \frac{1}{2}\mathbb{E}_{z\sim p(z)}\left[(D(G(z)) - 1)^2\right] \quad (5)$$

$$\ell_D = \tfrac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}}\left[(D(x) - 1)^2\right] + \frac{1}{2}\mathbb{E}_{z\sim p(z)}\left[(D(G(z)))^2\right] \quad (6)$$

We ran a few experiments by replacing minimax loss in the adversarial loss component of the base AC-GAN with the least squares loss, but noticed a slight degradation in the generated image quality. We decided to continue with the minimax loss for rest of the experiments.

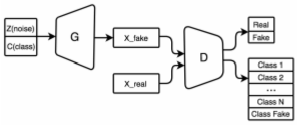

Figure 2. FCGAN Framework [17]

4.6. Fast-Converging Conditional GAN (FCGAN)

In [17], Li et al. introduce a simple yet very useful change to the ACGAN architecture for fast convergence. The idea is to add an additional class C_{fake} to the Auxiliary classifier and make the discriminator classify all the generated images as C_{fake} instead of mixing them up with real labels. Hence, the class loss for the discriminator gets changed as follows for FCGAN:

$$L_C^D = (E[logP(C = C_i|X_{real})] + E[logP(C = C_{fake}|X_{fake})]) \tag{7}$$

The architecture overview of FCGAN is shown in Figure-2.

4.7. Wasserstein GAN (WGAN)

To improve the stability of learning, and get rid of problems like mode collapse, Arjovsky et al. [9] proposed a new GAN variant (WGAN) that uses a cost function based on the earth mover's distance (EMD), which is a measure of the distance between two probability distributions. The exact technical details of this loss function are beyond the scope of this report. The key takeaway is that in WGAN, the discriminator network becomes a critic. It is very similar to the discriminator in a regular GAN, but it omits the final sigmoid function and outputs a scalar score rather than a probability. This score can be interpreted as how real the input images are.

Gulrajani et al. [14] observed that WGAN sometimes generates only low-quality samples or fails to converge. They found that these problems are often due to the use of weight clipping in WGAN to restrict the maximum value of the discriminator weights. They propose an alternative to clipping weights by penalizing the norm of gradient of the critic with respect to its input. This GAN variant is known WGAN-GP (WGAN with gradient penalty). It uses a ResNet architecture for both generator and critic.

We trained a model by combining the WGAN-GP loss function with the auxiliary loss function from AC-GAN, and using residual networks for both generator and discriminator. In the remainder of this report, this model is referred to as Conditional WGAN-GP or CWGAN.

5. Experiments and Results

5.1. Evaluation Metrics

We present a brief qualitative analysis of the images generated by our models, and use the Fréchet Inception Distance (FID) [15] for a quantitative evaluation. We used the trained generator model to generate a batch of images equal in number as the test (or validation) set, and then computed the FID score for these two sets of images. A lower FID score denotes better performance. We also use human evaluation to compare the images generated by different models.

For qualitative analysis, we find the nearest neighbor of the generated sample from the training set conditioned on the class. We estimate the nearest neighbors using $\mathbb{L}2$ norm in pixel space.

5.2. Hyperparameter Tuning

Due to the lack of time, we could not do extensive hyperparameter tuning, but below is a list of things we tried -

1. We tried both ReLU and LeakyReLU($\alpha = 0.2$) activations in the generator. LeakyReLU gave better results and we used it in both generator and discriminator networks of conditional DCGAN, ACGAN and FCGAN. We used Adam($l_r = 0.0002, b_1 = 0.5, b_2 = 0.999$) optimizer to train both networks across these models. We tried SGD($l_r = 0.001$) for the generator of FCGAN but it did not work better and due to the lack of time, we could not try other learning rates. For CWGAN, we used Adam($l_r = 0.0001, b_1 = 0, b_2 = 0.9$) as recommended by Gulrajani et al. [14]

2. We tried different batch sizes - $\{32, 64, 128\}$ for FCGAN and $\{64, 128\}$ for ACGAN and the FID plots of this experiment are shown in Figure-3. We notice that batch size=64 works best for FCGAN and batch size=128 for ACGAN. We also notice that ACGAN is much more sensitive to batch size change than FCGAN. We suspect this is because ACGAN is more sensitive to noisy batches. We tried different latent vector dimensions - $\{100, 128, 256\}$ for FCGAN and the FID plots of this experiment are shown in Figure-4. Latent dimension = 100 turns out to be the best and we used it across all our models.

5.3. Overfitting

Our training data, especially for genres - $\{$Computers, Science$\}$, has a lot of very similar cover designs across several books. Figure-5 gives one such example. This was causing our models to overfit for these classes by generating these repetitive designs seen in the training data. (See

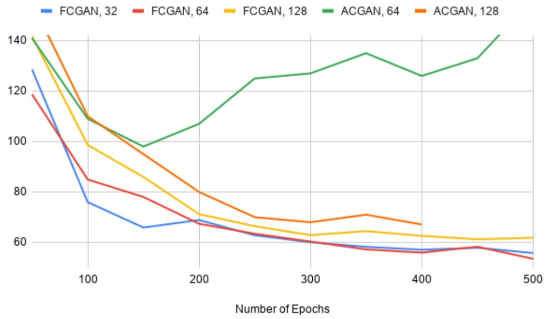

Figure 3. FID Curve obtained by training FCGAN and ACGAN over 2 classes {Children's, Computers}. We kept all hyper-parameters constant other than batch size.

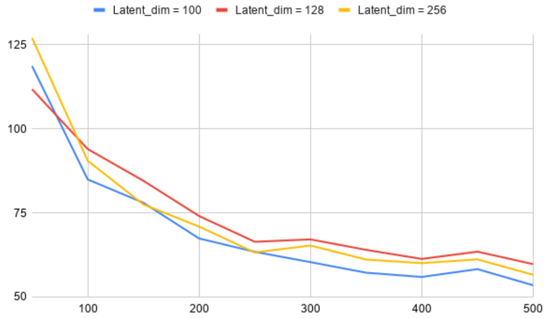

Figure 4. FID Curve obtained by training FCGAN over 2 classes {Children's, Computers}. We kept all hyper-parameters constant other than latent vector dimension.

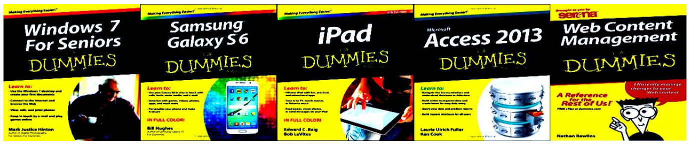

Figure 5. Design repetition in Computers class training data.

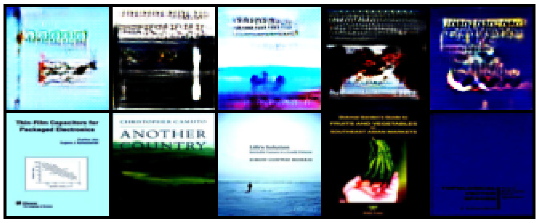

Figure 6. Generated images (top row) of class {Science} by FCGAN when trained for classes - {Children's, Science} compared with their nearest neighbors (bottom row).

figures-6, 7). To mitigate this, we left out these two classes for the final run of our models.

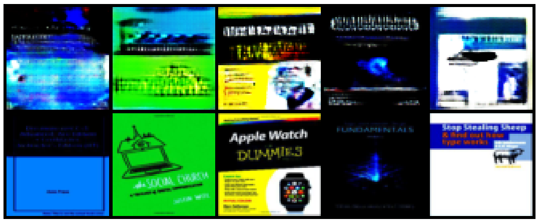

Figure 7. Generated images (top row) of class {Computers} by FCGAN when trained for classes - {Computers, Fiction} compared with their nearest neighbors (bottom row).

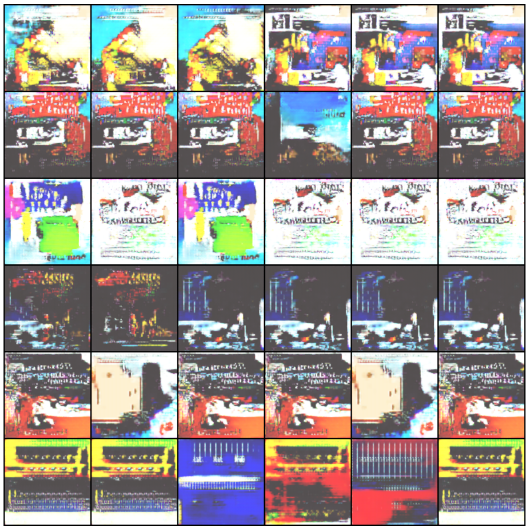

Figure 8. Mode Collapse of ACGAN model prior to real label smoothing and input normalization. Each row represents a class in the same order as Table-1

5.4. Mode Collapse & ACGAN Experiments

Mode collapse is when the generator generates a limited diversity of samples, or even the same sample, regardless of the input. We were facing this problem in the early versions of ACGAN and FCGAN (see Figure-8).

To overcome the problem of mode collapse, we ran the experiments listed in Table-2. With a combination of LeakyRelu for activation, one-sided label smoothing of real labels and noisy label inputs to the discriminator, we were able to overcome the problem and improve our model's performance significantly.

5.5. ACGAN vs FCGAN

We used human evaluation to compare the generated images of ACGAN and FCGAN and concluded that FCGAN performed slightly better, especially in producing real looking images (see Figure-9). This is expected as the auxilary loss of FCGAN does not let the generator fool the discriminator easily by explicitly maintaining a C_{fake} class, thus enabling a faster convergence of the model to a better re-

Adversarial Loss	Activation	Real Label Smoothing	Noisy label	Image Size	FID
Binary Cross Entropy	ReLU	None	$p = 0$	128x128	Mode Collapse
Wasserstein loss	LeakyReLU	{low=0.75, high=1.0}	$p = 0.05$	64x64	120.54
Binary Cross Entropy	ReLU	{low=0.75, high=1.0}	$p = 0.05$	128x128	106.41
Binary Cross Entropy	LeakyReLU	{low=0.75, high=1.0}	$p = 0.05$	128x128	86.88
LS Loss	LeakyReLU	{low=0.75, high=1.0}	$p = 0.05$	64x64	76.42
Binary Cross Entropy	LeakyReLU	{low=0.75, high=1.0}	$p = 0.05$	64x64	67.10

Table 2. Validation FIDs across different versions of ACGAN when trained on the classes - {Children's, Cookbooks}. The hyper-parameters are fixed to be {Batch Size, Latent Vector Dimension, Number of Epochs} = {128, 100, 400}

Genre	Baseline DCGAN		FCGAN	
	Val	Test	Val	Test
Children's	65.02	52.36	77.54	62.13
Cookbooks	80.66	60.57	94.50	75.56
Fiction	70.58	52.46	73.03	55.60
Travel	93.56	81.50	57.76	45.58
All 4	38.93	34.16	39.50	34.39

Table 3. FIDs of baseline DCGAN as compared with FCGAN trained for individual classes.

sult.

5.6. Conditional WGAN-GP

Our Conditional WGAN-GP model produced by far the best results in terms of image quality. But training the model for just 2-classes took over 4 days for 650 epochs on a {8 vCPUs, 52 GB memory, 1 x NVIDIA Tesla T4} VM and hence, we could not afford to experiment more with this model. From the 2-class results, we notice that for a creative genre like {Children's}, the model produces very good images as shown in Figure-10. But, for a more specific genre like {Cookbooks}, the model is not producing relevant images as shown in Figure-11. CWGAN's training loss plot is shown in Figure-12.

5.7. FCGAN

Due to the lack of time and resources to experiment more with CWGAN, we present FCGAN as our best model. Table-3 shows how it compares with baseline DCGAN model and Figures-13, 14 show their respective generated images. FCGAN's training loss plot is shown in Figure-19.

FCGAN not only produces images comparable with baseline, it is able to learn the design features of different genres very well. For example, the model is able to learn i) placement of title and food images on the cover as the design for Cookbooks (See (Figure-15)), ii) to place cartoon images at the center for Children's design cover (See Figure-16), iii) placement of landscape images on the cover as the design for Travel (See Figure-17), and iv) large font, dark colored images for Fiction (See Figure-18).

Model	Parameters, Train Time	FID
Conditional DCGAN	{128, 100, 400}, 4 hrs	82.23
ACGAN w/ LS Loss	{128, 100, 400}, 4 hrs	76.42
ACGAN	{128, 100, 400}, 4 hrs	67.10
CDCGAN w/ LS Loss	{128, 100, 400}, 4 hrs	62.04
FCGAN w/ LS Loss	{64, 100, 500}, 5 hrs	61.34
Baseline DCGAN	{128, 100, 500}, 5 hrs	56.37
FCGAN	{64, 100, 500}, 5 hrs	55.40
Conditional WGAN-GP	{64, 100, 650}, 96 hrs	48.23

Table 4. Validation FIDs across the best versions of different models when trained on the classes - {Children's, Cookbooks}. The hyper-parameters are in the order - {Batch Size, Latent Vector Dimension, Number of Epochs}.Training time as measured on the best VM we had access to: {8 vCPUs, 52 GB memory, 1 x NVIDIA Tesla T4}

5.8. Overall Results

Table-4 sums up the results across all the different models we tried on the 2 genres of {Children's, Cookbooks}. We show images generated by our best Conditional DCGAN model (which uses Least Squares Loss) in Figure-21 and our best ACGAN model in Figure-20. While FCGAN performs slightly better than the baseline DCGAN, Conditional WGAN-GP beats it by a significant margin. The major difference in the performance of FCGAN and Conditional WGAN-GP comes from the quality of the images the two models generate. Even though ACGAN, FCGAN are very promising in learning different genres' unique features, they still suffer greatly in image quality when compared with the images generated by Conditional WGAN-GP.

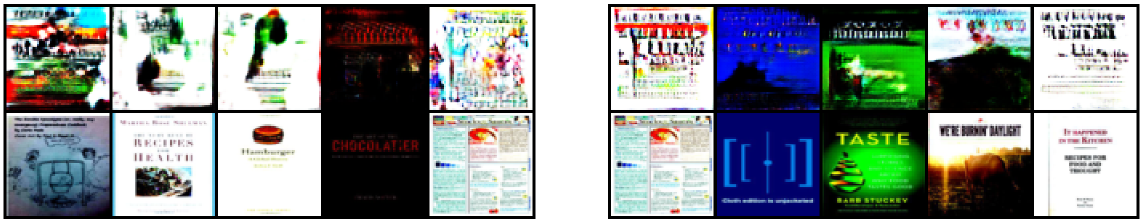

Figure 9. Side by side comparison of images generated by best versions of ACGAN (left) vs FCGAN (right) for the class {Cookbook}

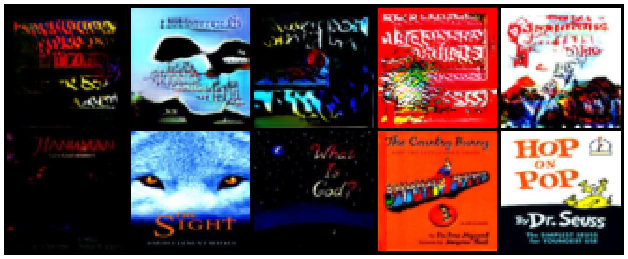

Figure 10. Generated images (top row) of class {Children's} by our best CWGAN when trained for classes - {Children's, Cookbooks} compared with their nearest neighbors (bottom row).

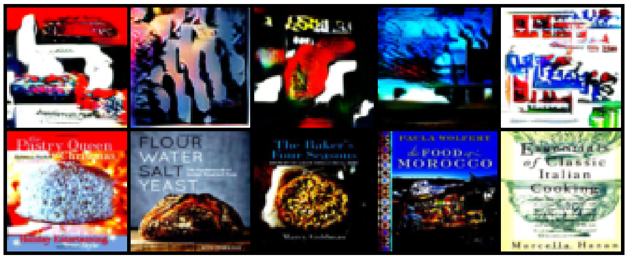

Figure 11. Generated images (top row) of class {Cookbooks} by our best CWGAN when trained for classes - {Children's, Cookbooks} compared with their nearest neighbors (bottom row).

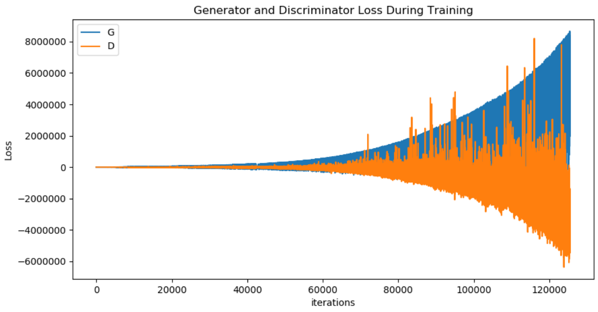

Figure 12. Training Loss Plot of CWGAN when trained over the classes - {Children's, Cookbooks}

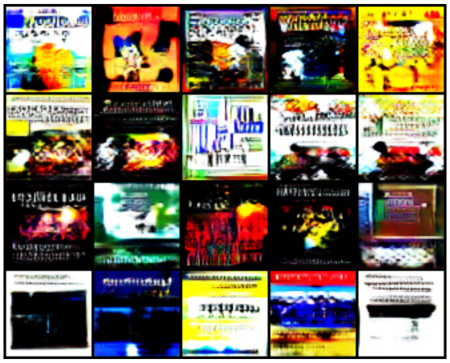

Figure 13. Images generated by our best FCGAN model when trained for the classes - {Children's, Cookbooks, Fiction, Travel} with full data available for each class as given in Table-1. Each row represents a class in the order - {Children's, Cookbooks, Fiction, Travel} from top to bottom.

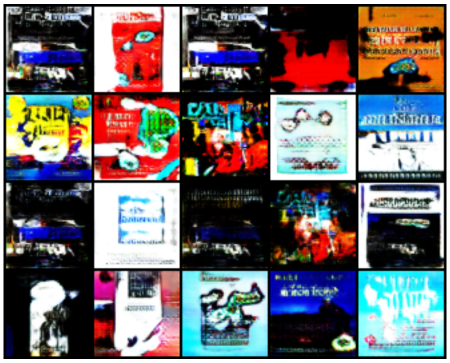

Figure 14. Images generated by baseline DCGAN model when trained with data of classes - {Children's, Cookbooks, Fiction, Travel}. Note: The images are not conditioned on genre.

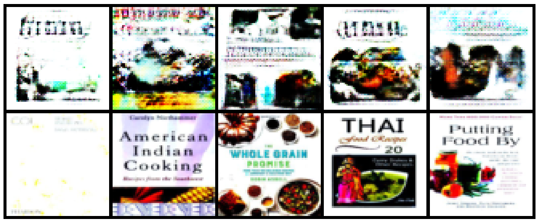

Figure 15. Generated images (top row) of class {Cookbook} by FCGAN when trained for that class alone compared with their nearest neighbors (bottom row).

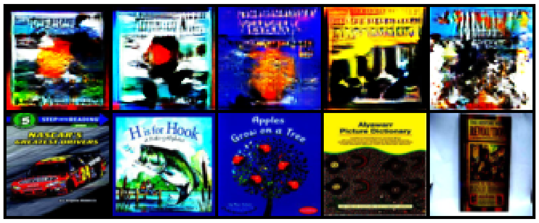

Figure 16. Generated images (top row) of class {Children's} by our best FCGAN when trained for classes - {Children's, Cookbooks, Fiction, Travel} compared with their nearest neighbors (bottom row).

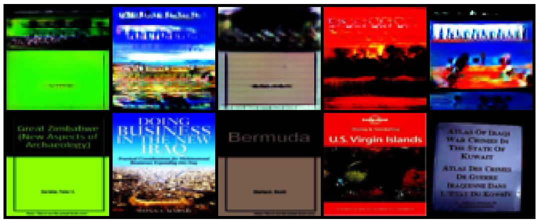
Great Zimbabwe (New Aspects of Archaeology)
DOING BUSINESS IN THE NEW IRAQ
Bermuda
U.S. Virgin Islands
ATLAS OF IRAQI WAR CRIMES IN THE STATE OF KUWAIT
ATLAS DES CRIMES DE GUERRE IRAQUIENNE DANS L'ETAT DU KOWEIT

Figure 17. Generated images (top row) of class {Travel} by FCGAN when trained for classes - {Children's, Travel} compared with their nearest neighbors (bottom row).

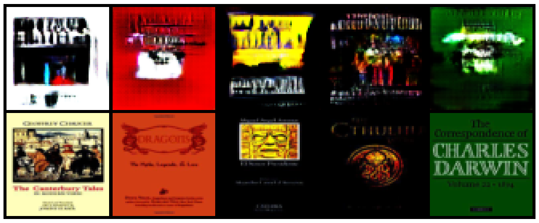

Figure 18. Generated images (top row) of class {Fiction} by our best FCGAN when trained for classes - {Children's, Cookbooks, Fiction, Travel} compared with their nearest neighbors (bottom row).

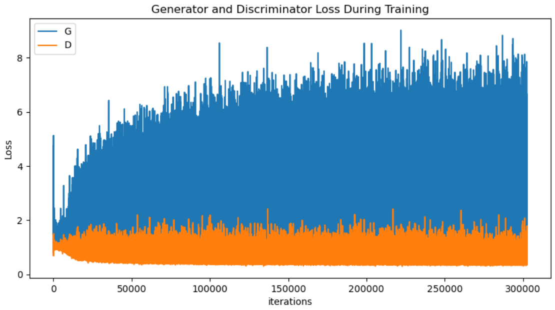

Figure 19. Training Loss Plot of FCGAN when trained over the classes - {Children's, Cookbooks, Fiction, Travel}

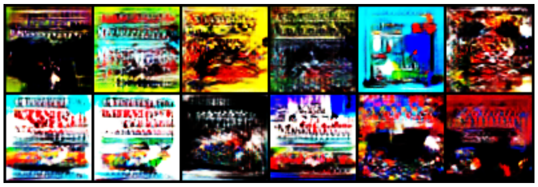

Figure 20. Images generated by our best ACGAN model when trained for the classes - {Children's, Cookbooks}. Each row represents a class in the order - {Children's, Cookbooks} from top to bottom.

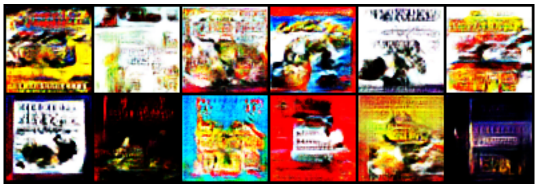

Figure 21. Images generated by our best Conditional DCGAN model (with Least Square Loss in place of adversarial loss) when trained for the classes - {Children's, Cookbooks}. Each row represents a class in the order - {Children's, Cookbooks} from top to bottom.

6. Conclusion & Future Work

In summary, to generate book cover designs conditioned upon different genres, ACGAN, FCGAN and Conditional WGAN-GP are very promising models. Our ACGAN and FCGAN models show they are able to learn the unique design of different genres but struggle with image quality, whereas Conditional WGAN-GP excels really well in producing high quality images but suffers from understanding the nuances that distinguish different classes. Due to lack of resources, we could not do further experiments with Conditional WGAN-GP model.

Improving the quality of the generated book cover designs and making them look more realistic requires more work and research. Below is a list of experiments that should be worth trying out:

- Experiment more with Conditional WGAN-GP. As a first step, train the model for a longer duration to try and make it learn the nuances of different genres better.

- Experiment with more data. We identified Kaggle datasets ([8], [21]) with around 33K and 27K images that can be supplemented to our current dataset. From our analysis so far, we observe that more data enhances the model performance.

- Do more extensive hyperparameter tuning. Currently, we have not tried to optimize the generator optimizer, discriminator optimizer, learning rates, architecture components, etc. So there is a lot of scope for optimization that can be done by extensive hyperparameter tuning using cross-validation.

7. Acknowledgements

For FID implementation, we used the code available at [6]. Our DCGAN baseline model implementation is based on the official Pytorch DCGAN tutorial [5]. For implementing conditional WGAN-GP, we used the code available at [4]. For implementing all other variants, we referred to a public github repository that contains implementations of multiple GAN variants [7].

We would like to extend the warmest gratitude to the entire teaching staff of CS231N for not only a wonderfully enriching learning experience, but also for their thoughtful and empathetic considerations during these challenging times. We would also like to thank Professor Ranjay Krishna and CS231N TA Chris Waites for their valuable inputs and feedback on the project.

8. Appendix

8.1. Appendix A

Figure-22 [23] shows the DCGAN generator architecture

8.2. Appendix B

Table 5 summarizes the hyperparameters used in the design of the AC-GAN and FC-GAN generators.

Table 6 summarizes the hyperparameters used in the design of the AC-GAN and FC-GAN discriminators.

8.3. Appendix C

To understand how FCGAN works with different genres and to finalize the list of genres for training our final model, we trained FCGAN on every combination of 1-class and 2-classes. The results of this experiment are presented in Table-7.

We also trained our 4-class FCGAN model by keeping the data distribution equal among all the classes to check if skewed data sizes were causing any hindrance to learning. We noted that that wasn't the case and more data always seems to enhance the model's performance. The generated images from this training are given in Figure-25.

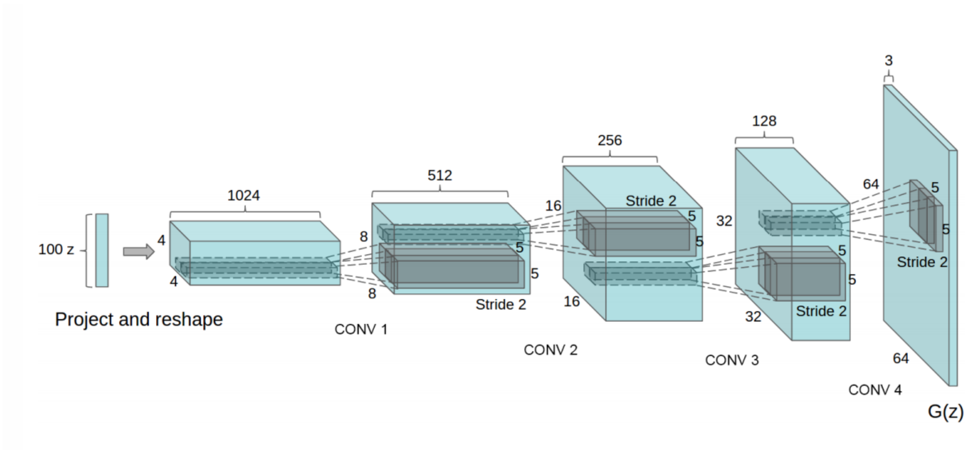

Figure 22. DCGAN Architecture [23]

Operation	Kernel	Strides	Feature Maps	BN?	Dropout	Nonlinearity
$G(z)$ - $(100 + N_{classes}) \times 1 \times 1$ input						
Linear	N/A	N/A	768	✗	0.0	None
Transposed Convolution	4×4	1×1	384	✓	0.0	LeakyReLU
Transposed Convolution	4×4	2×2	192	✓	0.0	LeakyReLU
Transposed Convolution	4×4	2×2	96	✓	0.0	LeakyReLU
Transposed Convolution	4×4	2×2	48	✓	0.0	LeakyReLU
Transposed Convolution	4×4	2×2	3	✗	0.0	Tanh

Table 5. AC-GAN and FC-GAN Generator hyperparameters. $N_{classes}$ refers to the number of class labels, and z is 100 dimensional gaussian noise

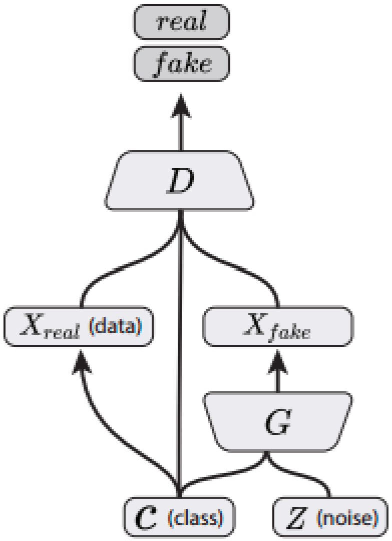

Figure 23. Conditional GAN Design [20]

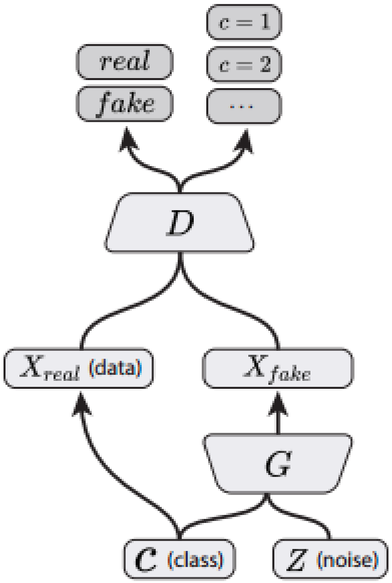

Figure 24. AC-GAN Design [22]

Operation	Kernel	Strides	Feature Maps	BN?	Dropout	Nonlinearity
$D(x)$ – $64 \times 64 \times 3$ input						
Convolution	3×3	2×2	16	✗	0.5	LeakyReLU
Convolution	3×3	1×1	32	✓	0.5	LeakyReLU
Convolution	3×3	2×2	64	✓	0.5	LeakyReLU
Convolution	3×3	1×1	128	✓	0.5	LeakyReLU
Convolution	3×3	2×2	256	✓	0.5	LeakyReLU
Convolution	3×3	1×1	512	✓	0.5	LeakyReLU
Linear	N/A	N/A	1 + $N_{classes}$	✗	0.0	Soft-Sigmoid

Table 6. AC-GAN and FC-GAN Discriminator hyperparameters. $N_{classes}$ refers to the number of class labels

	Children's	Computers	Cookbooks	Fiction	Science	Travel
Children's	77.54	58.37	55.40	53.07	55.67	44.83
Computers	58.37	88.96	65.43	57.59	64.70	48.52
Cookbooks	55.40	65.43	94.50	57.91	61.69	58.20
Fiction	53.07	57.59	57.91	73.03	62.91	51.23
Science	55.67	64.70	61.69	62.91	88.08	53.12
Travel	44.83	48.52	58.20	51.23	53.12	57.76

Table 7. Validation FIDs of FCGAN trained over 2-classes or single class.

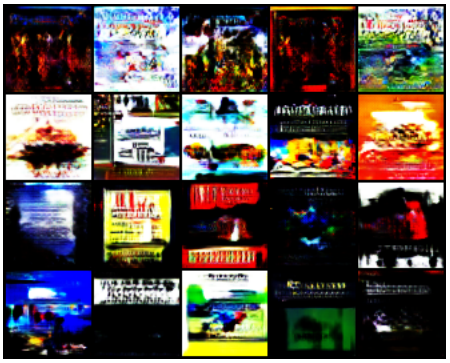

Figure 25. Images generated by our best FCGAN model when trained for the classes - {Children's, Cookbooks, Fiction, Travel} with equal data distribution of $Train + Val = 7014$ per class. Each row represents a class in the order - {Children's, Cookbooks, Fiction, Travel} from top to bottom.

9. References

References

[1] Book cover design:your 7-step guide to an unforgettable cover. https://blog.reedsy.com/book-cover-design/.

[2] How to design a great book cover. https://scribewriting.com/book-cover-design/.

[3] Project code. https://github.com/ankitdwivedi23/cs231n-project.

[4] Pytorch cwgan implementation. https://github.com/jalola/improved-wgan-pytorch.

[5] Pytorch dcgan tutorial. https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html.

[6] Pytorch fid implementation. https://github.com/mseitzer/pytorch-fid.

[7] Pytorch gan implementations. https://github.com/eriklindernoren/PyTorch-GAN.

[8] Luka Anicin. Kaggle book covers dataset. https://www.kaggle.com/lukaanicin/book-covers-dataset, 2020.

[9] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[10] Cristian Bodnar. Text to image synthesis using generative adversarial networks. *CoRR*, abs/1805.00676, 2018.

[11] Przemysław Buczkowski, Antoni Sobkowicz, and Marek Kozlowski. Deep learning approaches towards book covers classification. 01 2018.

[12] Ian J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160, 2017.

[13] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[14] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. *CoRR*, abs/1704.00028, 2017.

[15] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, abs/1706.08500, 2017.

[16] Brian Kenji Iwana and Seiichi Uchida. Judging a book by its cover. *CoRR*, abs/1610.09204, 2016.

[17] Chengcheng Li, Zi Wang, and Hairong Qi. Fast-converging conditional generative adversarial networks for image synthesis, 2018.

[18] Suraj Maharjan, Manuel Montes, Fabio A. González, and Thamar Solorio. A genre-aware attention model to improve the likability prediction of books. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3381–3391, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics.

[19] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, and Zhen Wang. Multi-class generative adversarial networks with the L2 loss function. *CoRR*, abs/1611.04076, 2016.

[20] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[21] Naren. Kaggle goodreads best books ever dataset. https://www.kaggle.com/meetnaren/goodreads-best-books/version/3, 2018.

[22] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans, 2016.

[23] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.

[24] Scott E. Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *CoRR*, abs/1605.05396, 2016.

[25] Martin Arjovsky Michael Mathieu Soumith Chintala, Emily Denton. Gan hacks. https://github.com/soumith/ganhacks.

[26] Wei Ren Tan, Chee Seng Chan, Hernán E. Aguirre, and Kiyoshi Tanaka. Artgan: Artwork synthesis with conditional categorial gans. *CoRR*, abs/1702.03410, 2017.

[27] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris N. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *CoRR*, abs/1612.03242, 2016.