Yu-GAN-Oh: A Novel Application of DCGAN

1st Mikhail Kardash
Department of ECE
UCSD
San Diego, CA
mkardash@eng.ucsd.edu

2nd Kenny Situ
Department of ECE
UCSD
San Diego, CA
ksitu@eng.ucsd.edu

3rd Austin Choe
Department of ECE
UCSD
San Diego, CA
achoe@eng.ucsd.edu

4th Albert Tan
Department of ECE
UCSD
San Diego, CA
aktan@eng.ucsd.edu

***Abstract*—Generative Adversarial Networks (GANs) can be applied to many tasks in image generation due to their dual-network structure which pits a generator network against a discriminator network to produce synthetic images resembling real images. We use a specific class of GANs known as Deep Convolutional GANs (DCGANs) to produce original images of Yu-Gi-Oh cards from a dataset of 10,000 images. Techniques for improving model performance such as feature matching and mini-batch discrimination are also explored. Model outputs resemble the structure of desirable images, but the resolution is poor. Furthermore, the model is susceptible to mode collapse, in which the weights are trapped in a local minimum, rendering the model unable to generate a greater variety of images. This is somewhat alleviated by implementing advanced techniques to improve GAN performance.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

The unsupervised task of generating data from a distribution has become an increasingly important area of research. GANs accomplish this by the structure of their feedback loop between the generator and discriminator, where both networks attempt to optimize opposing loss functions in a zero-sum game. The discriminator is a CNN which classifies images as real or fake, while the generator is a deconvolutional CNN which generates images.

The YGOPro2 image set, which consists of about 10,000 images of Yu-Gi-Oh cards, is used to train our network. Based on the probability distribution of trading cards, the GAN produces samples of generated data that we compare with the true data.

Various numerical metrics exist; however, they do not fully capture the essential features that are needed to determine the validity of generated cards. Therefore, our model is evaluated through visual inspection of the outputs.

II. Motivation

The continued development of image generation via neural networks has achieved remarkable feats in the fields of super resolution, denoising, and even creating original artistic images. However, state-of-the-art algorithms still struggle to generate images with high levels of structure while maintaining variety of content. We seek to create a model capable of preserving the structural details from a set of images such as a frame and text box while learning to construct high variance, original, artistic renderings.

The trading card game, Yu-Gi-Oh! offers a collection of highly detailed images with well defined content including a name, attribute, type, and effect all within its carefully framed structure. We believe if our model is capable of efficiently recreating this design with original content, our algorithm could even be extended to 3D generation tasks with voxels for robotics simulation environment construction.

III. Methods

In this section, we elaborate upon our novel Yu-Gi-Oh! card generation network with the following components:

A. Generator

The Generator network architecture takes in a random sequence of length 100 and constructs an image of dimension 428x321x3. The layers consist of five sets of consecutive 2D Transpose Convolution, Batch Normalization, and ReLU. Finally, we apply one more 2D Transpose Convolution and a Tanh activation function. As opposed to traditional CNNs, the DCGAN replaces maxpooling functions with fractionally-strided convolutions, which allows the network to learn its own spatial upsampling. Batch Normalization is used after each transpose convolutional layer to reduce covariance shift so that gradients flow through the network. Note that no fully connected layers are used in order to increase model stability.

B. Discriminator

The Discriminator network performs binary classification to determine the validity of the Generator output. The layers consist of 2D Convolution and Leaky ReLU, followed by three sets of 2D Convolution, Batch Normalization, and Leaky ReLU. Then we apply one more 2D Convolution followed by two Fully Connected (FC) layers that encode the features of the image and determine the most significant features, respectively. A learned Transformation function is applied to the encoded features as a tensor multiplication. This Transformation operation offers an alternative to pixel-to-pixel comparison as a metric of similarity, which will be further discussed in the next subsection. Finally, the sigmoid activation function is applied to the feature encodings to output the logit.

Fig. 1. Example of each card category from left to right, top to bottom: trap card, spell card, normal monster, effect monster, fusion monster, ritual monster, synchro monster, xyz monster, pendulum monster, and link monster

Fig. 2. Example of trap card with features in colored boxes: cyan: card name, yellow: attribute, red: card category (spell or trap), blue: card art, green: card effect.

C. Dataset

In Figure 1, we provide an example of various cards with distinct characteristics used in our data set. Note the color of the frame of the card indicates different types of cards; spell and trap cards are green and magenta, respectively, while monster cards take on a variety of colored frames. Figures 2 and 3 show the various features to be learned by the Discriminator in determining the validity of a real Yu-Gi-Oh! card, and we expect to see these features present in our Generator outputs.

D. Training and Innovations

To train the GAN, we hold one network constant while training the other. This is to ensure that the discriminator weights are unaffected while training the generator, and vice versa. The workflow of the training loop is as follows: train the discriminator to classify between real and fake by using both ground-truth images as well as synthesized images from the generator. Then, set the discriminator as untrainable and train the generator by passing its outputs into the discriminator. Perform backpropogation so that the generator improves the quality of the images it produces across epochs. Repeat until the specified number of epochs is reached, or when the discriminator becomes unsure of every generated image and guesses real or fake roughly half the time. We first resized the images and center cropped to get 64x64 results. Once these were satisfactory, we used similar hyperparameters to train a model using full-size images.

Fig. 3. Example of monster card with features in colored boxes: cyan: card name, yellow: attribute, red: level, blue: card art, green: card type, effect, and status.

One problem experienced by deep networks is overconfidence, which results in using very few features to classify images. In a GAN, the Generator may learn which specific features the Discriminator is using to perform classification and exploit this to produce only those features. To overcome this greedy optimization, we implement one-sided label smoothing. This technique penalizes the certainty of the Discriminator by reducing the logit value by a random fraction, up to $1/5$.

Another issue that frequently plague GANs is mode collapse, in which outputs start to exhibit unusually high similarity. We implement minibatch discrimination to help mitigate this problem. This technique involves computing a dissimilarity score for each image in a batch by comparing it to other images in that batch via the learned Transformation function. These scores are calculated after the first FC layer and appended to the input of the second FC layer. Thus, this dissimilarity metric becomes another feature activation where

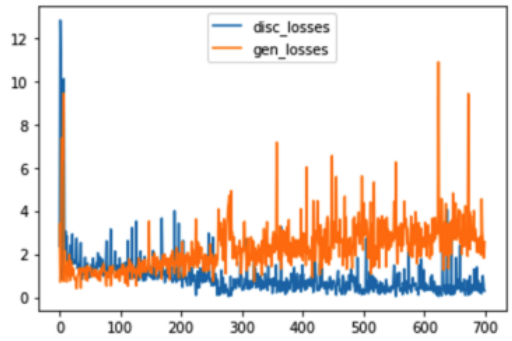

Fig. 4. Training loss for the baseline DCGAN Generator and Discriminator networks.

higher scores encourage the Generator to learn more from images with little resemblance to one another.

Perhaps the most pertinent issue in training GANs is when the model fails to converge. This occurs when the Generator and Discriminator adversarial optimization becomes too greedy, and the Generator's "best" image to fool the Discriminator constantly changes. The source of this problem is the Generator objective of beating the Discriminator so we may promote convergence by proposing an additional, alternative objective function. Our new objective function is then defined as follows:

$$||\mathbb{E}_{\mathbf{x}\sim p_{data}}\mathbf{f}(\mathbf{x}) - \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{z}}(\mathbf{z})}\mathbf{f}(G(\mathbf{z}))||_2^2 \quad (1)$$

where $\mathbf{f}(\mathbf{x})$ is the feature vector input to the first FC layer. This expands the goal of the Generator from simply beating the Discriminator to creating images with features similar to those of real images.

IV. Results

In this section we present the results of our experiment and compare outputs from a baseline DCGAN with our improved Yu-GAN-Oh.

A. Baseline DCGAN

We implemented baseline DCGAN as a preliminary architecture to generate Yu-Gi-Oh! cards, but the model yielded poor results. In Figure 5, we provide an example output to assess the model's performance. The image is very dark and exhibits significant noise with few discernible features. The frame is brownish orange in color, indicating a monster card, but the card name and attribute is not visible. The level seems to be present but not countable. There is a text box for the card effect and status, but they are illegible. The art in the card is an assortment of colors without a distinguishable creature, object, or character. Thus, this model clearly failed to generate a feasible Yu-Gi-Oh! card.

B. Yu-GAN-Oh

Initial implementation of Yu-GAN-Oh yielded clear indication of mode collapse, in which all outputs exhibited highly similar card features with low variance in the images. We only

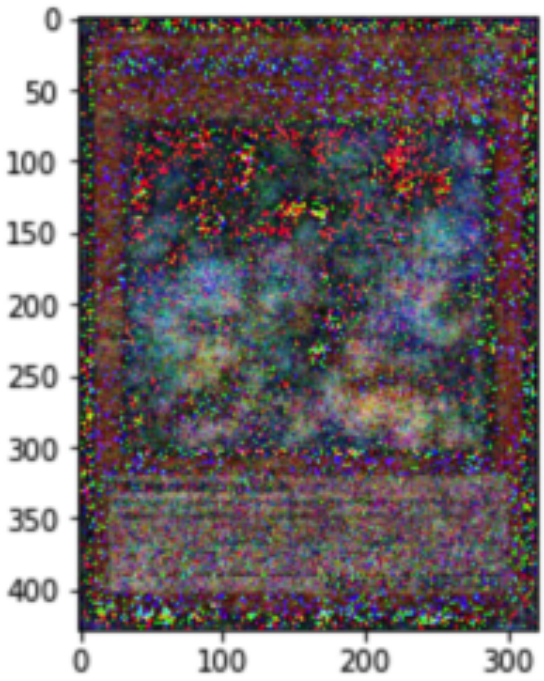

Fig. 5. Example output image from baseline DCGAN.

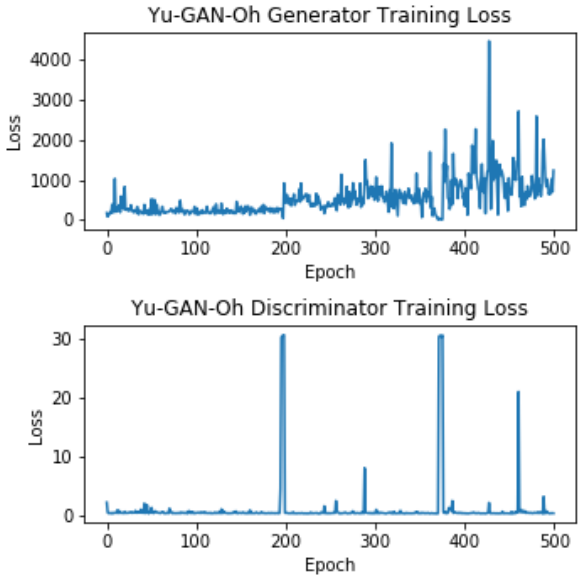

Fig. 6. Training loss for the Yu-GAN-Oh Generator and Discriminator networks.

Fig. 7. Set of nine example outputs from Yu-GAN-Oh without mini-batch discrimination.

Fig. 8. Set of nine example outputs from Yu-GAN-Oh with mini-batch discrimination. Note that a fusion monster is present in the last row, second column.

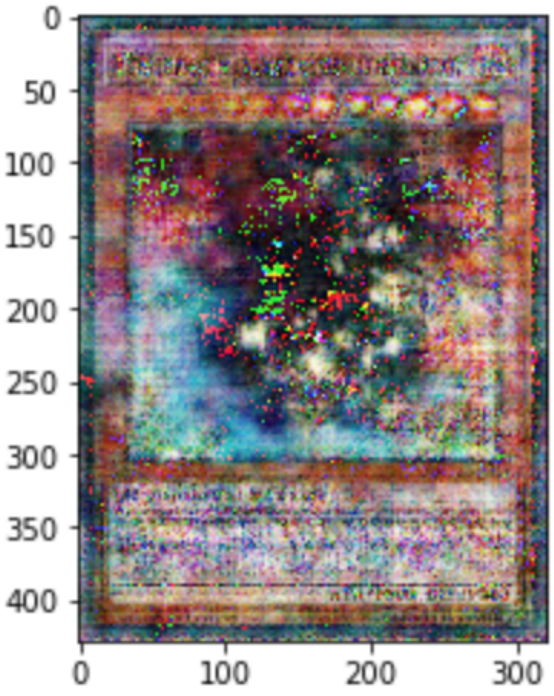

Fig. 9. Example output from Yu-GAN-Oh.

observed standard effect monster cards with the same level and very similar images. By implementing one-sided label smoothing and feature matching, the GAN output a greater variety of features such as the presence of spell and trap cards as seen in Figure 7. However, the monster cards still exhibited high feature similarity. They all appear to have three stars and the same attribute.

By implementing mini-batch discrimination, the results improved significantly. Observing the sample images in Figure 8, we notice a greater variety of level stars and higher variation in the images. Furthermore, spell and trap cards no longer exhibit the brownish orange card frame color that indicates a monster card. We notice that the model even managed to produce a fusion monster, which is indicated by the purple card frame. The high variance in output images clearly shows that this implementation successfully overcomes the issue of mode collapse.

Comparing the results of Yu-GAN-Oh output image in Figure 9 with the output image of the baseline DCGAN model from Figure 5, we notice a substantial increase in the clarity of the card. Although the card name is still not legible, some form of text is clearly present. The attribute and levels are also highly visible; the number of stars clearly indicates a level eight monster. The text box clearly show fields for the monster type, effect, and status located at the top left, main body, and bottom right of the text box, respectively. Unfortunately, the card art is still a formless assortment of colors that does not seem to resemble any creature, character, or object. Despite the significant improvements in the generated output, a significant amount of noise is still present, and we may also observe square artifacts from the convolutional layers.

V. Conclusions and Future Improvements

VI. Individual Contributions

References

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use “Ref. [3]” or “reference [3]” except at the beginning of a sentence: “Reference [3] was the first . . .”

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors’ names; do not use “et al.”. Papers that have not been published, even if they have been submitted for publication, should be cited as “unpublished” [4]. Papers that have been accepted for publication should be cited as “in press” [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

References

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.
[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
[3] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
[4] K. Elissa, “Title of paper if known,” unpublished.
[5] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
[7] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove the template text from your paper may result in your paper not being published.