Visual Search by Brushing

Baldo Faieta, Mingyang Ling, Yifu Wang
Adobe
{bfaieta,mling,yiwang}@adobe.com

Abstract

In this work we investigate a type of visual search which is guided by the user and is well suited to work with a mobile touch screen. The concept is for the user to first specify a category of images to narrow the search domain and then either start with a query image or with a blank canvas and progressively modify it with visual editor brushing operations. The modified query image is continuously being reinterpreted to correspond to plausible images in the corpus within the specific category. As the user modifies the query image, similar images in the corpus are being presented and at any point the user can swap the query image by one of the candidate results. We use interactive Adversarial Generative Networks (GAN) to generate the images and use the generated images as a query for image similarity. The trained models are conditional on the text query and we explore the results with the Oxford-102 Flower dataset as well as Adobe Stock sample dataset.

1. Introduction

As of now, billions of people carry phones with cameras in their pockets and they take over a trillion photos every year [13]. Eventually, all these photos are stored and indexed in the cloud. In this context, there is a need for powerful tools to search not only through a growing set of personal collections but on public repositories as well as licensed visual media services. As more and more people use their phones with touch interfaces to access and search these collections, we need to adapt our traditional keyword-based user interfaces for search to be more mobile and touch friendly. Visual search, which allows users to find similar images based on a query image [16], is a very good alternative but the user needs to have an example of the instance they need to search for this type of search to work. Even with a query picture, trying to describe similar images having additional constraints (e.g., I like an image with this bird, but with longer beak and with red feathers) is difficult and even more so using a mobile user interface.

In this work we investigate a particular type of visual search which is guided by the user and it is well suited to work with a touch screen. The concept is for the user to first specify a category of images to narrow the search domain and then either start with a query image or with a blank canvas and progressively modify it with visual editor brushing operations. The modified query image is continuously being reinterpreted to correspond to plausible images in the corpus within the specific category. As the user modifies the query image, similar images in the corpus are being presented and at any point the user can swap the query image by one of the candidate results.

Generative adversarial networks [14] (GANs) provide us with a mechanism for generating natural images that conform with a trained set within a corpus. They are trained by having two networks, one that generates candidate images based on a latent variable (usually sampled from the uniform distribution) and another that discriminates whether an image has been generated (fake) or is part of the training set (real). There are countless variations, on this initial setup [9], [15], [24], [18], [25]. In particular, Generative Adversarial Text to Image Synthesis GANs (GAN-CLS) [22] give us a technique to focus the generated image conforming to a specific text. However, these GANs do not provide a mechanism to modify the generated image interactively. Another set of interactive GANs, iGAN [17] and the Neural Photo Editor [6], enable us to interactively modify the image being generated.

In order to generate images interactively based on the user input, our model adopts the iGAN model described in [17], where an image x is first translated to z in the latent space domain and then, in this domain, z is progressively modified to conform to the constraint editing (brushing) operations by the user. iGAN was trained to generate just one class of images and in this work we focus on supporting any class of images described by a text query. For this, we adapt GAN-CLS to work interactively like the iGAN model to generate images conforming to a text query and modified interatively by the user. The resulting generated images then are being continuously used to find similar images in the main corpus.

2. Background

Our work is related to both models that focus on interactively generating images as well as generating images narrowed to a class or to a text description. In this section we would like to review both types of models.

2.1. Interactive GANs

As mentioned in the previous section, we follow closely the iGAN model proposed by [17], which is similar to the Introspective Adversarial Network (IAN) described in [6] for their Neural Photo Editor. In [17], a GAN G is trained to generate images x in an ideal low dimensional manifold X of natural images given a z in a latent space domain Z. For a given image x_0 in X, an inverse model is also trained to project x_0 to z_0 in Z.

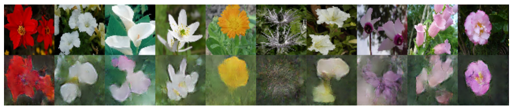


Figure 1: Original sample images from the Oxford-102 flower dataset translated to latent space and then regenerated back using iGAN

Figure 1 shows the results of using the inverse model trained by iGAN to project images to the latent space Z and regenerate them back using the corresponding GAN.

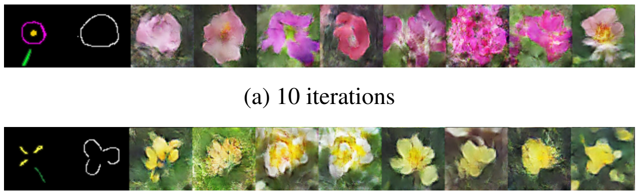
The brush operations g by the user are casted as constraints $f_g(x) = v_g$ on a local part of an image x. Then, a new z_1 is found such that the corresponding image $G(z_1)$ incorporates the constraints specified by the user. Using G, z_1 is found using gradient descent minimizing the following equation:

$$z^* = arg \min_{z \in Z} \sum_g \|f_g(G(z)) - v_g\|^2 + \lambda_s \|z - z_0\|^2 \quad (1)$$

In equation (1) above, the first term in the sum incorporates the constraints (e.g., color brushing operations) while the second term corresponds to a manifold smoothness that enforces moving in small steps in the manifold so as not to alter the original image x_0 too much. Deep Convolutional Generative Adversarial Networks (DCGAN) [10] is used as the underlying GAN, however it is suggested that this component could be swapped by other any of more powerful models.

Figure 2 shows examples of generated images $G(z_1)$ by iGAN trained on the Oxford-102 dataset after applying color and edge operations (the first 2 cells in the sequence).

Even though $G(z_1)$ captures the changes the user wants in the image, [17] use a dense correspondence algorithm to adjust the original photo to produce a more photo-realistic



(a) 10 iterations

(b) 100 iterations

Figure 2: Generated results after user applying brush operations. First cell correspond to color brush. Second cell corresponds to edge brush. The rest 8 cells correspond to generated images.

result. In contrast, [6] for the Neural Photo Editor, uses a masking technique to transfer the reconstruction changes to the original image. In this work, we use $G(z_1)$ as is given that we use the image as the query for image similarity and this step is more forgiving to degradation issues.

2.2. Text and Class Conditional GANs

Conditional GANs [20], [11] provide a mechanism to focus the generated images specific to a particular class. The basic idea is to feed the generator and the discriminator with class labels so that the generator can produce samples conditioned on a class. An alternative method is to task the discriminator to reconstruct the class information [23]. Auxiliary classifier GAN (AC-GAN) [9] propose a variant that leverage both approaches where the model is class conditional with an auxiliary decoder tasked to reconstruct the labels. AC-GAN was trained on ImageNet data on 1000 of the categories. However, [9] reports mode dropping and recommended dividing up the 1000 ImageNet classes across 100 AC-GANs for training stability. In this work, we tried AC-GAN with our dataset and we failed to get good results. Moreover, given the search use case ideally we prefer to deal with a single model as well as a broader choice of categories for the generated images.

There is another class of work focused on synthesizing images based on a text description. Text conditioned auxiliary classifier generative adversarial network (TAC-GAN) [8] builds upon AC-GAN conditioning the the generated images on a text description instead of a class label. In this case, the input to the generator is both the noise z and an embedding representation for the text. TAC-GAN was trained with the Oxford-102 dataset [21] using Skip-Thought vectors to generate the text embeddings from the image captions. Similar to AC-GAN the loss used is both a real/fake discriminator as well as a discriminator for the class the image belongs to. The text embeddings are just used as auxiliary information.

The work of [22] for generative adversarial text to image synthesis (GAN-CLS) also use text to synthesize images.

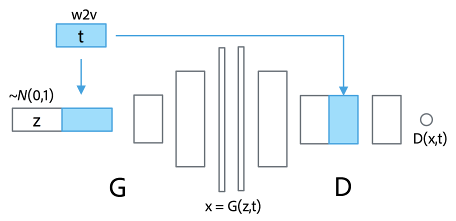


Figure 3: GAN-CLS text-conditional GAN architecture taken from [22]. The text embedding t is used both in the generator G to generate the image $x = G(z, t)$ and for discriminator D to distinguish whether the image x is fake or real.

The GAN architecture of GAN-CLS, as seen in Figure 3 is very similar to TAC-GAN in that the generator receives a concatenated vector of a text embedding and the noise vector z. in contrast to TAC-GAN though, the discriminator has to judge whether the pair (text, image) is real or fake. During training, the loss function of the discriminator includes scores for (real image, right text), (real image, wrong text) and (fake image, right text) pairs. In this work, we use this same algorithm and report encouraging results. [22] reports increase of diversity in the within-class generated images by using interpolations between embedding pairs to train the model. As we will discuss further, we did not find that is the case in our experiments with our datasets.

2.3. GAN Inverse

As explained above, both iGAN and the Neural Photo Editor apply the editing constraints by the user in the latent space domain Z and therefore a mechanism to project potentially an initial image x into the latent space is needed. In the case of iGAN, [17] found that a combination of a projection by optimization as well as learning a feed-forward network that predicts the latent z vector works best. [7] describes a method similar to iGAN projection by optimization by finding the z that when passed through the generator produce images that are visually similar to x.

Another approach is to learn two models in parallel, one that given an image x can give you a value z in the latent space and another that learns the inverse mapping where given a latent value z' can generate back an image x' is described in [15] with their Bidirectional Generative Adversarial Network (BiGAN) and in [24] with their Adversarial Learned Inference (ALI) model. The BiGAN and ALI models are more general cases of the ones used by iGAN and IAN.

In this work, we opted to use the projection by optimization approach for simplicity sake though we have not evaluated the penalty in terms of degradation that results from just using this method.

3. Methods

In order for an interactive generative visual similarity system to work, we need several components that have to work in concert. Figure 4 outlines the various steps needed and where each of these components enter into play. As explained in the previous section, our model uses a text-conditional GAN which relies on feeding text embeddings together with a noise vector to the GAN in order to generate candidate images. Therefore as first step (1), we need to be able to derive these text embeddings t from a given text. For this, we use pre-trained tag embeddings for images trained using the mutual information between tags based on tag co-occurrence [12]. When a user inputs the text to narrow the search, we find the corresponding text (tag) embeddings and add them up to form one embedding t (though we expect the user to just input a single word/tag).

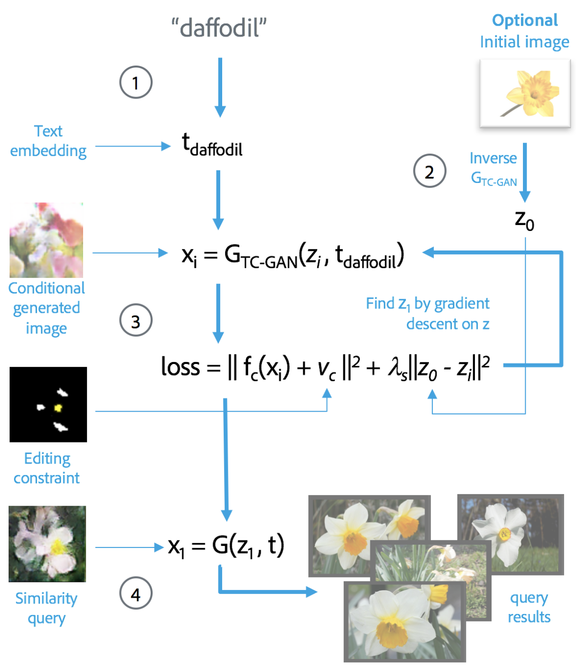


Figure 4: flowchart outlining the algorithm used to generate a query conditional on a text description and incorporating the interactive constraints of the user

The next step (2) is to use an optional initial image x_0 that the user may have selected. The system can generate images without it, but it could be that the user wants to start with a given image. In this case, we need an inverse GAN to find the corresponding z_0, as described in the previous section. In this work we use the projection-by-optimization method as described in [7]. Basically, we start with a noise vector z and apply gradient descent based on a simple L1 loss function that compares $x0$ to $G(z)$.

The following step (3) is to find $G(z_1, t)$ such that it conforms to the text represented by the text embedding t. In this case, we use the text conditional GAN-CLS [22] as the GAN that generates the images conditional on the text embedding. We use then gradient descent on z and we use equation (1) as the loss function. The outcome $G(z_1, t)$ is a generated image that we can then use as query for image similarity [16] as part of step (4) where the most similar images are those with the largest cosine distance. [17] uses both color and edge constraints. Color constraints are calculated by applying a mask to the candidate image and comparing it with the brush colors within the mask using an L2 norm. The edge constraint is calculated using the HOG metric on the candidate image and comparing it with the edge mask.

3.1. GAN-CLS

To train GAN-CLS we follow the algorithm proposed in [22]. That is, first we derive h and $\hat{h}$ corresponding to the matching and mis-matching text. Then we generate a fake image $\hat{x} = G(z, h)$ using a noise vector z sampled from $N(0, 1)$. Using the discriminator, we calculate the scores
$s_r = D(x, h)$ for the real image, right text pair,
$s_r = D(x, \hat{h})$, for the real image wrong text pair and
$s_f = D(\hat{x}, h)$ for the fake image, right text pair.
With these scores we compute the loss for the discriminator D and the generator G:

$$\begin{aligned}\mathfrak{L}_D &= log(s_r) + \frac{1}{2}(log(1 - s_w) + log(1 - s_f)) \\ \mathfrak{L}_G &= log(s_f)\end{aligned}$$

In terms of the network architecture, the generator G has a fully-connected layer that compresses the text embedding t and then concatenated with the noise vector z. Following this we use a traditional deconvolutional network using ReLU as the non-linearity and using batch normalization after every convolutional layer. For the discriminator D, we use strided convolutions followed by batch normalization and for the non-linearity layer we use leaky ReLU.

4. Experiments

4.1. iGAN baseline

In order to validate iGAN, we first trained a DCGAN model [10] using the Oxford-102 Flower dataset [21]. This dataset contains 8189 images divided up in 102 classes. Figure 5 shows a sample of the original images as well as generated images after epoch 200. Using this model, with the iGAN model we generated a set of images that conform to a given color and edge constraints. The results of this experiment can be seen in Figure 2. Notice that there is no starting image. This experiment is the result of just using iGAN as originally constructed in [17] and we can see that the quality of the results are very reasonable.

(a) real samples

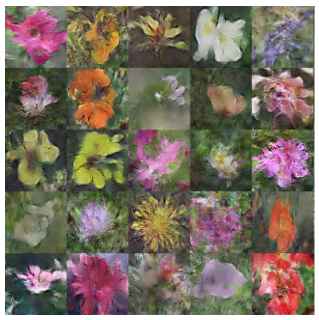

(b) Generated images for epoch 200

Figure 5: Real vs. generated images using DCGAN model on Oxford-102 dataset

(a) real samples

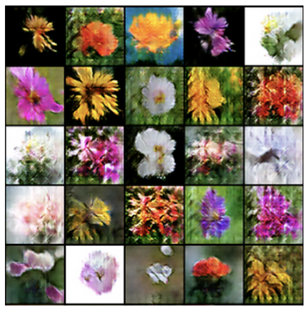

(b) Generated images for epoch 74

Figure 6: Real vs. generated images using GAN-CLS model on Adobe Stock flower dataset

4.2. GAN-CLS

To train the text-conditional GAN-CLS, we generated a dataset that is congruent with the Oxford-102 dataset. For this, we took a sample of 9965 images from Adobe Stock [1], an image licensing service, out of a set of 59 million images. These images are annotated with on average of 52 tags out of a vocabulary of 39965 tags. These tags where used to embed both the images and the tags in an embedding with 2048 dimensions using the technique described in [12] (see above). The images we selected were the top ranking set that included the tag ”flower” and had a square size resized to be of size 64x64x3. Figure 6a shows a sample of these images. We use the pre-trained embedding associated with every image to train the GAN-CLS. Notice that every image in the Adobe Stock dataset has a pre-trained embedding and every tag has a pre-trained embedding and both belong to the same embedding by design.

To train the GAN-CLS, we used a learning rate of 0.0002, the size of the noise vector of 100, the size of the compressed text vector of 128. We used Adam optimization with $\beta_1 = 0.5$, $\beta_2 = 0.999$. Figure 6b shows generated images after epoch 74 of one of the training runs. Notice that this generated images are based on random text embeddings associated with real images in the dataset.

In Figure 7, we show the results of generating images with the trained GAN-CLS by combining the associated text embedding for the text ”chrysanthemum”, ”daffodil”, ”daisy”, ”rose”, and ”sunflower” with a a random noise. Two points to notice is that first, the random noise does not generate diversity within the class as it seems that the model is getting mode dropping within the class. Second, is that the even though the classes are similar to the actual types of flowers, there are still details that are not accounted for in the generated images like for example the color of the daisies in Figure 7c and the size of the sunflower inner floret in Figure 7e.

4.3. Combining iGAN with GAN-CLS

In order to evaluate the degree to which we can modify an image of a specific category (i.e., an image that has a tag with the category name as label) based on editing constraints, we took a generated image specific to a category and manually modified the color constraints. Figure 8 shows 2 examples for the classes ”chrysanthemum” and ”daffodil”. Notice that the edge constraints constraint the results well within the boundaries of the expected image. This manual test allow us to see what we can expect if we just use iGAN.

However, the real challenge is to use the method described in Section 3 to generate images specific to a category while conforming to editing constraints by the user. Figure 9 shows the results of generating an image algorithmically from the ”daffodil” category using color brushing constraints. Notice that the first initial image being generated is of a yellow daffodil, yet the image is being progressively modified to white conforming to the color constraints.

Figure 10 shows the results of looking for similar images to the image in the last cell of Figure 9. The point of this would be that if a user just looks with the keyword ”daffodil” or ”white daffodil”, she would get yellow daffodil images. In this way, the use would be adding an additional constraint to be similar to the white daffodil image just generated.

Figure 11 shows additional examples of generating images specific to text categories while conforming to color editing brushing constraints. Notice how in Figure 11b the initial generated image is a red rose and because of the yellow editing constraint, it becomes yellow, while in Figure 11c is the other way around using the same mask with a red color instead. Notice also how in contrast to the the generated images from iGAN in Figure 8, these images do not have edge constraints and therefore presumably it is easier to derail from the selected class.

5. Discussion

In Figure 12 we show generated images specific to the ”rose” category for models saved at particular epochs. We can readily see that within category there is mode dropping. That is, all the diversity of the images drops the more epochs we train. Another way of representing this is by showing the diversity as calculated by the MS-SSIM metric [19].

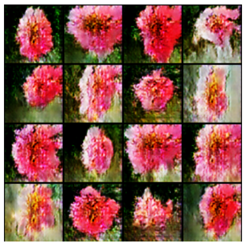
In Figure 13 we plot the MS-SSIM score for images generated from models trained with increasing number of epochs specific to the category ”rose”. As we can see in the plot, in earlier epochs, there is more diversity but after about 60 epochs the diversity goes down and stays this way. This happens for most of the categories we’ve looked into. [22] proposes training with interpolations between embedding pairs to mitigate this loss within class diversity. However, with our datasets, we found that this technique did not produce much different results.

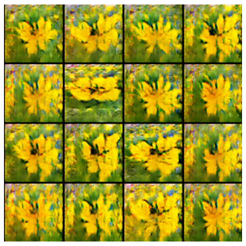
There seems to be a trade off between loosing diversity versus increasing quality of the generated images the more epoch training there is. In order to evaluate this, we sampled generated images specific to the category ”rose” for models corresponding to increasing number of epochs and for these images we queried the original Adobe Stock corpus of 59 million images to find similar images. We picked the top 200 results of each query and aggregated the counts of tags found. We sorted the tags by order of popularity and picked the top 10 tags. If the tag ”rose” was present, we counted as 1. The plot in Figure 14 shows the percentage of queries per epoch that have the tag ”rose”. As we can see in the plot, the quality improves progressively up to a point. However, beyond some point over 100 epochs the quality starts to go down. Figure 14 and Figure 13 together show this trade off.

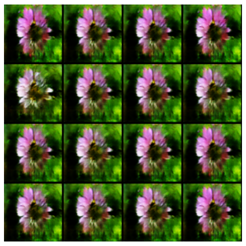
6. Conclusion

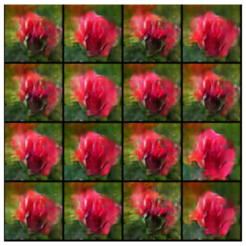
In this work we have shown that by using text conditional GAN we can potentially generate a wider variety of images and even though text conditional GANs tend to have mode dropping within categories, by interactively adding editing constraints to the image generation, the user can force the models to generate different category instances conforming to the user constraints. However text conditional GANs still seem to show mode dropping and even though in this particular application, the interactive constraints may mitigate it, there still seems to be that further research needs to happen to improve text conditional GANs.

In this work we have tried to extend the concept of iGAN by being able to limit the generative images for specific categories. There are other directions also worth pursuing like being able to paste partial images onto the canvas as constraints for the generator to take into account. Also, be-


(a) chrysanthemum


(b) daffodil


(c) daisy


(d) rose

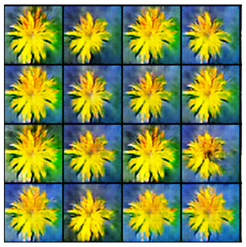
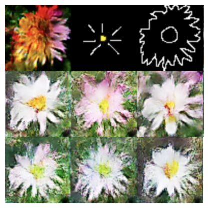

(e) sunflower

Figure 7: Generated images for GAN-CLS trained on Adobe Stock flower dataset where the text embedding of a specific flower name is passed as one of the parameters to the generator


(a) category chrysanthemum

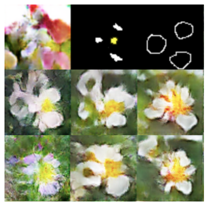

(b) category daffodil

Figure 8: Using iGAN to manually edit images generated by GAN-CLS specific to a category. The top corner image is the image generated by GAN-CLS specific to a given text embedding. The next top cells in the row are both the color and edge constraints. The next 6 cells are images generated by iGAN

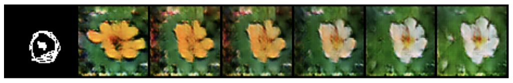


Figure 9: Using algorithm from Section 3 to generate an image for the category **”daffodil”** using brushing constraints. The first cell corresponds to the brushing colors. The rest are the images being generated at steps 0, 20, 40, 60, 80, 100 of the gradient descent loop

ing able to generate partial images by the user guiding the model where to generate into the canvas are potential extensions to iGAN.

The application of iGAN to generate a query for similarity is a use case that is requires further investigation as this may be a very fruitful interaction paradigm for users to search in touch mobile user interfaces.

Acknowledgements

We would like to thank Eli Shechtman and Chen Fang at Adobe Research and Jun-Yan Zhu at U.C. Berkeley for useful suggestions to guide this work. Also, we would like to thank Zhe Lin at Adobe Research for providing us with model for the text embeddings to the Adobe Stock images.

To train and evaluate many of the iGAN examples, we used the code at [3] The code that applies the color constraints is based on [3]. [4] as well as [5] were used as references to create the pytorch code to run the GAN-CLS. [2] also was used as inspiration for part of the pytorch code to apply the editing constraints.

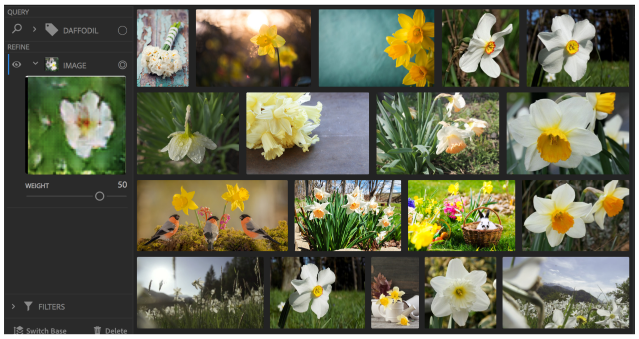
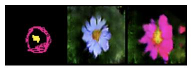
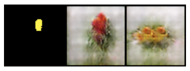


Figure 10: Image similarity based on the daffodil generated image conforming with to the white brush constraints. The results come from a corpus of 59 million images and the recall set is narrowed to include images that have the tag ’daffodil’


(a) category **daisy**


(b) category **rose**

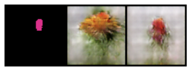
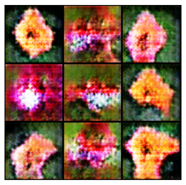

(c) category **rose**

Figure 11: Images generated specific to a category and conforming to editing constraints. The first cell correspond to the color brushing constraints, the second cell the initial image generated and the last cell corresponds to the generated image after 100 steps in the gradient descent phase.

References

[1] Adobe stock. `http://stock.adobe.com/`.

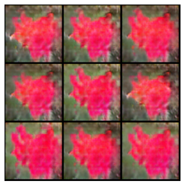
[2] Cs231n assignment 3. `http://cs231n.github.io/assignments2017/assignment3/`, 2017.

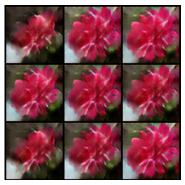

(a) epoch 20

(b) epoch 40

(c) epoch 60

(d) epoch 80


(e) epoch 100


(f) epoch 125

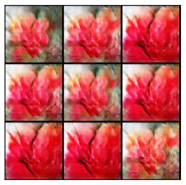

(g) epoch 150

Figure 12: Generated images specific to the category **rose** for models corresponding to training for various epochs

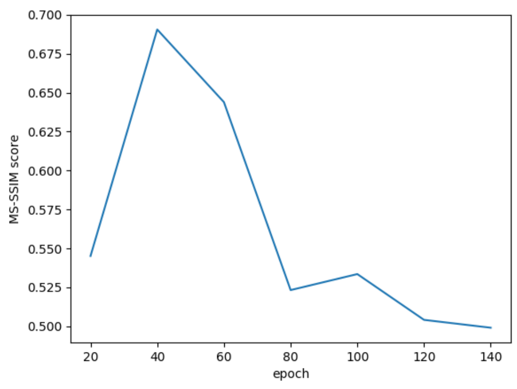


Figure 13: Average MS-SSIM score calculated among generated images sampled for the category ”rose” for models saved at particular epochs

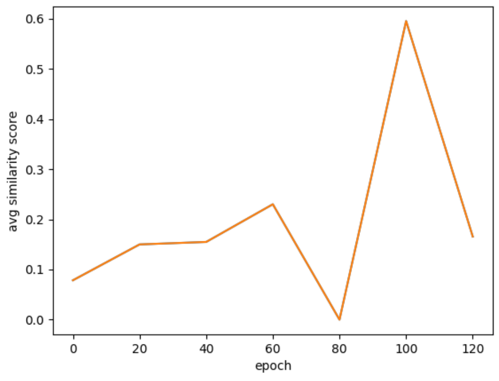


Figure 14: Average MS-SSIM score calculated among generated images sampled for the category ”rose” for models trained for various epochs

[3] junyanz/igan github. `https://github.com/junyanz/iGAN`, 2017.

[4] reedscot/icml2016 github. `https://github.com/reedscot/icml2016`, 2017.

[5] wiseodd/generative-models github. `https://github.com/wiseodd/generative-models`, 2017.

[6] J. R. N. W. A. Brock, T. Lim. Neural photo editing with introspective adversarial networks. *NIPS*, 2016.

[7] A. A. B. A. Creswell. Inverting the generator of a generative adversarial network. Arxiv preprint, 2016.

[8] S. A. M. L. A. Dash, J. Gamboa and M. Z. Afzal. Tac-gan - text conditioned auxiliary classifier generative adversarial network. Arxiv preprint 1703.06412, 2017.

[9] J. S. A. Odena, C. Olah. Conditional image synthesis with auxiliary classifier gans. *ICLR*, 2017.

[10] S. C. A. Radford, L. Metz. Unsupervised representation learning with deep convolutional generative adversarial networks. *ICLR16*, 2016.

[11] O. V. L. E.-A. G. K. K. A. van den Oord, N. Kalchbrenner. Conditional image generation with pixelcnn decoders. Arxiv preprint 1606.053228, 2016.

[12] F. Chollet. Information-theoretical label embeddings for large-scale image classification. Arxiv preprint 1607.05691.

[13] S. Heyman. Photos, photos everywhere. *New York Times*, July 2015.

[14] M. M. B. X. D. W.-F. S. O. A. C. Y. B. I. Goodfellow, J.Pouget-Abadie. Generative adversarial nets. *NIPS*, 2014.

[15] T. D. J. Donahue, P. Krahenbuhl. Adversarial feature learning. *ICLR*, 2017.

[16] T. L. C. R. J. W.-J. P. B. C. Y. W. J. Wang, Yang Song. Learning fine-grained image similarity with deep ranking. *CVPR*, 2014.

[17] E. S. A. A. E. J.-Y. Zhu, P. Krahenbuhl. Generative visual manipulation on the natural image manifold. *ECCV*, 2016.

[18] P. I. A. A. E. J.-Y. Zhu, T. Park. Unpaired image-to-image translation using cycle-consistent adversarial networks. Arxiv preprint 1703.10593, 2017.

[19] Z. W. Z. D. H. Y.-H. L. K. Ma, Q. Wu and L. Zhang. Group mad competition - a new methodology to compare objective image quality models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[20] S. O. M. Mirza. Conditional generative adversarial nets. Arxiv preprint 1411.1784, 2014.

[21] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[22] X. Y. L. L. B. S.-H. L. S. Reed, Z. Akata. Generative adversarial text to image synthesis. In *Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA*, volume 48 of *JMLR*, 2016.

[23] W. Z. V. C. A. R. T. Salimans, I. Goodfellow and X. Chen. Improved techniques for training gans. Arxiv e-prints, 2016.

[24] B. P. O. M. A. L.-M. A. A. C. V. Dumoulin, I. Belghazi. Adversarial learned inference. *ICLR*, 2017.

[25] H. X. R. Y. K. L.-Z. W. X. Mao, Q. Li. Least squares generative adversarial networks. Arxiv preprint 1611.04076, 2016.