Progressive Growing in Color Space for Training Generative Adversarial Networks

Daniel Huang[1], Oliver Brady[1]

Abstract

Training generative adversarial networks is challenging due to slow training speed and instability. Prior studies have demonstrated that progressive growing of image dimensions can make these issues less severe. In our study, we propose a novel method of progressively growing the number of unique colors in an image. We perform color quantization using unsupervised k-means clustering. We then train a CycleGAN model for an image translation task. We start training on images with few colors, and then gradually increase the number of unique colors in the image. By conducting quantitative and qualitative analysis on the outputs, we find that our approach accelerates the training time of CycleGAN to generate realistic images.

1. Introduction

Generative adversarial networks (GANs), though very powerful, are notoriously difficult and unstable to train. Lots of current work with GANs deals with how to improve training so that GANs either produce better outputs quicker, or can train longer without losing stability. One proposed solution is progressive growing, which has demonstrated improvements for training speed and stability. Prior studies have only progressively grown the image dimensions.

In our project, we propose a novel approach to overcome these issues, using k-means color clustering to progressively grow the number of unique colors in each image during the training of a GAN. In other words, at the beginning of training, the generator is inputed images with only two colors, uses that input to generate images, and the discriminator classifies those images as real or fake. As training progresses, we gradually increase the number of unique colors.

The input to our algorithm is two image datasets, each with different styles. An example that we use is one dataset of summer images, and one of winter. Our algorithm then color quantizes the images, meaning it reduces the number of colors in the image, with an unsupervised k-means algorithm. We then train CycleGAN, a GAN built for style transfer, on this newly processed dataset, changing the amount of colors per image based on epoch. Thus, the output of our model is generator weights that take an image of one style as an input, and transfer the style for the output image.

2. Related work

2.1. Progressive Growing

Progressive growing is a training methodology whereby the generator and discriminator both grow progressively: the generator starts by generating low resolution images for the discriminator to process (Karras et al., 2017). As training progresses, new layers are added to the model (increasing model complexity) so that the networks can generate and discriminate images of increased dimensions. This modification accelerates training and provides greater stability. The authors are able to produce images from the CelebA dataset at resolution 1024 by 1024 (prior studies could only produce images at resolution 128 by 128). They also beat the state of the art for inception score in unsupervised CIFAR10 by 10%, indicating higher quality generated images.

2.2. Image Translation

Image translation is a class of computer vision tasks to translate an input image into a corresponding output image. Traditionally, this task has been tackled by building each part of the translation pipeline by hand and manually extracting features (Efros & Freeman, 2001). We investigate two seminal papers, Pix2Pix and CycleGAN. The main distinction in usage is that Pix2Pix requires a paired dataset of images from two different domains (for example, paired before and after photos from home renovations) (Isola et al., 2017). However, CycleGAN takes in two unpaired datasets (for example, a set of winter time landscape images and a set of summer time landscape images) (Zhu et al., 2017). Pix2Pix uses conditional adversarial networks for image-to-image translation. The networks learn the mapping from an input image to the corresponding output image, and also

[1]Stanford University

learns a loss function to train the mapping. For CycleGAN, there are two sets of generators and discriminators, one for each direction of translation, and cycle consistency over the two sets of GANs is enforced. Therefore, there is no need for paired images.

2.3. Color Quantization

Color quantization is a technique whereby the number of colors in an image is reduced, while still attempting to keep the image as close as possible to the original. It can be thought of as similar to dimensionality reduction techniques. Prior work has employed k-means clustering to perform color quantization to reduce a full-color image with 17 million different colors to a reduced set of 256 colors (Kasuga et al., 2000). We will employ this technique in our own investigation: as an alternative to growing image dimensions, we will instead grow the number of unique colors in each image. In order to generate images with different numbers of unique colors, we will use a variety of K values, where K is equal to the number of clusters, and therefore the number of colors in the image.

3. Dataset and Features

3.1. Home Renovation Dataset

We scraped 4,000 color images of before and after home renovation images (including bedrooms, bathrooms, kitchens, and living rooms), to create a paired image dataset of 2,000 image pairs. We scraped this data from Instagram using the HGTV hashtag as our search criteria and finding good before and after accounts. This dataset was used to build generative models (DCGAN, WGAN-GP), which are described in the project milestone.

3.2. Fireplaces Dataset

We scraped 300 color images of fireplaces (a variety including gas and electric fireplaces with different hearth designs and materials) from Google Images. This dataset was used to build generative models (DCGAN, WGAN-GP), which are described in the project milestone.

3.3. Mountains Dataset

We used the GeoPose3K dataset of 3,000 color images of mountains (Brejcha & Čadík, 2017). We create a paired dataset by running a Canny edge detector on each image to get the corresponding "sketch" of a mountain.

3.4. Yosemite Dataset

We downloaded the Yosemite dataset from CycleGANs GitHub (Zhu et al., 2017). There are 1273 images of Yosemite in the summer, and 854 photos from the winter. The CycleGAN team got the dataset by using the flickr API with date keys for summer and winter. We use a dataset with an 80% train/test split. This is the dataset we use to generate all of our final results.

4. Methods

4.1. Generative Models for Image Translation

Generative methods describe how a dataset could have been created, and are able to produce novel samples that could have been drawn from the underlying dataset. Approaches to generative models include autoregressive models, variational autoencoders (VAEs), and generative adversarial networks (GANs). A GAN is composed of a generator, which produces a possible fake data sample, and a discriminator, which distinguishes between real and fake samples. The two networks are trained in tandem in an adversarial fashion such that they will both improve over time. We tackle the image translation task, whereby the input is one image, and the GAN will "translate" the image into a different style, while preserving the content. For example, the GAN could "translate" a sketch of a mountain (with only edges) into a full-color picture of a mountain. We use two models, Pix2Pix and CycleGAN, which we have described in the Related Work section.

4.2. K-Means for Color Quantization

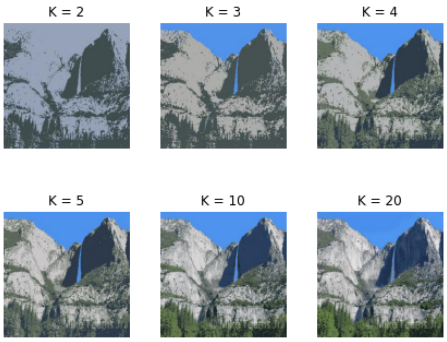

Figure 1. Examples of an image from the mountains dataset with different k-values (number of colors in an image)

K-Means is an unsupervised clustering algorithm which partitions n data points into k clusters. The assignment of points to a specific cluster depends on the distance to the centroid of that cluster, so k-means is a distance based algorithm.

For the purposes of our project, we use k-means to reduce the number of colors in the image (color quantization). We use the algorithm in Algorithm 1. The clustering is done in color space, so the distance between two points is a function of their RGB values, not their Euclidian location

Algorithm 1. Unsupervised K-Means

Input: data $x_0, ..., x_N$, desired number of clusters k
Output: set of k cluster centroids $C = \{c_1, ..., c_k\}$
Initialize: $C \leftarrow RandomSample(X, k)$
Repeat:
  Calculate new cluster for each sample:
  $S_i \leftarrow \{x : ||x - c_i||^2 \leq ||x - c_j||^2 \forall j, 1 \leq i, j \leq k\}$
  $S_i \wedge S_j = \emptyset$, each sample assigned to only one cluster
  Reassign samples to new clusters:
  $c_i \leftarrow \frac{1}{|S_i|} \sum_{x_j \in S_i} x_j$
Until: assignments S_i do not change

in the image. We treat the color space the same as three dimensional Euclidean coordinates, just using (R, G, B) Thus, the cluster assignment for a point is the color that is visibly most similar to that pixel. To quantize, a points color is replaced with the color of the mean of the cluster it is assigned too.

5. Experiments

For our milestone we trained the pix2pix GAN on the home renovation, fireplace, and mountain edge dataset. None of those outputs had an image resolution that we desired, so we moved onto CycleGAN. No longer needing a paired dataset, we trained on their Yosemite dataset of summer and winter photos.

We trained three separate CycleGAN models on the Yosemite dataset. The first model is our control; we trained the model as closely to the original paper as possible. Like the paper, we used a learning rate of 0.0002, with 9 residual blocks for the Generator and ReLU activation functions, and PatchGAN for the discriminator. We trained the GAN for only 20 epochs instead of the 200 in the paper because of a) compute power constraints and b) our project is a comparison of models, not about end performance, so as long as the epochs are consistent in all three, we are still able to make a comparison. The second and third models had the exact same architecture, but we changed the number of colors per image in the dataset that the model was training on as the epochs progressed.

The second model was trained on data that had been pre-processed by our k-means color quantize algorithm. The first epoch had input data where $K = 2$, the second had $K = 3$, third had $K = 4$, fourth had $K = 5$, fifth had $K = 10$, sixth had $K = 20$, and epochs 7 to 20 were run on the normal dataset. K is equal to the number of unique colors present in each image. We selected our values of K so that our model would learn the shapes and critical colors in the first few epochs, gradually learning more and more colors until the 7th epoch when the model is introduced to the un-processed dataset. The reason for the jumps in K from 5 to 10 and 10 to 20 is because as K increases, the difference in the output image decreases. An image quantized with $K = 2$ and $K = 3$ look radically different, but it is hard to tell $K = 20$ and $K = 21$ apart. Thus, the hope was that by jumping after $K = 5$, each epoch would continue to introduce helpful information.

The third model was also trained on color quantized data, this time the first five epochs were trained on a dataset where $K = 5$, the second five epoch were trained on a dataset where $K = 20$, and the last 10 were trained on the original dataset. We got the idea for this model by watching model 2 train. It seemed like $K = 1$ to $K = 4$ barely had enough color to help the GAN generate images. $K = 5$ seemed like a good balance of color and shape for the first couple epochs of training.

6. Results

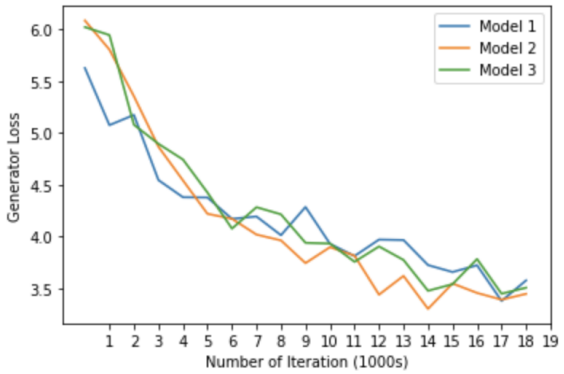

Figure 2. Generator Loss by iteration for models

Iterations vs. Generator Loss																		
Iterations (1000s):	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Model 1:	5.1	5.2	4.5	4.4	4.4	4.2	4.2	4.0	4.4	4.0	3.9	4.0	4.0	3.7	3.6	3.6	3.4	3.6
Model 2:	5.9	5.3	4.9	4.4	4.5	4.2	4.0	3.9	3.8	4.0	3.9	3.5	3.6	3.4	3.6	3.5	3.5	3.5
Model 3:	5.8	5.1	4.9	4.4	4.8	4.0	4.3	4.2	4.0	4.0	3.9	4.0	3.8	3.5	3.5	3.8	3.5	3.5

We have both quantitative and qualitative results from our training. For quantitative results, see Figure 2 for a graph of generator loss on all three models for each $1,000$ iterations.

Generator loss in CycleGAN has is made up of two parts: adversarial loss and cycle consistancy loss. Adversarial loss penalizes the generator if the discriminator is able to discern if the equation is real or fake. For a generator that maps, $G : X \rightarrow Y$, and its discriminator D_Y, we get:

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = E_{y \sim p_{\text{data}}(y)}[\log D_Y(y)]$$
$$+E_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))]$$

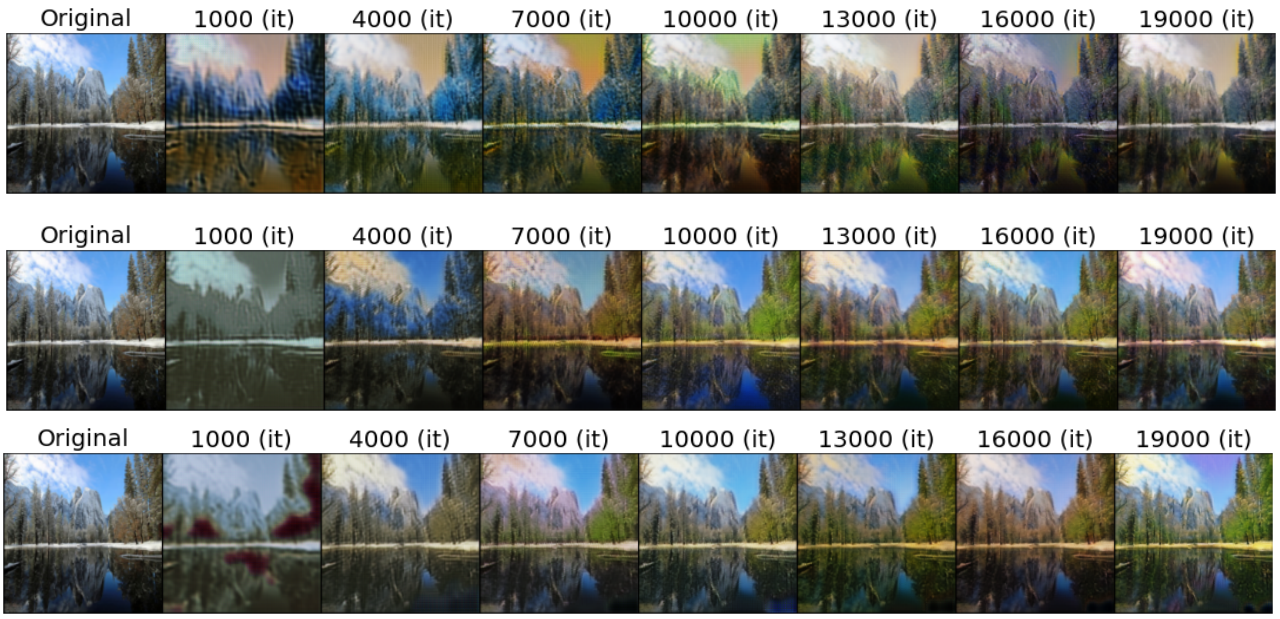

Figure 3. Model 1, 2, and 3 in respective order

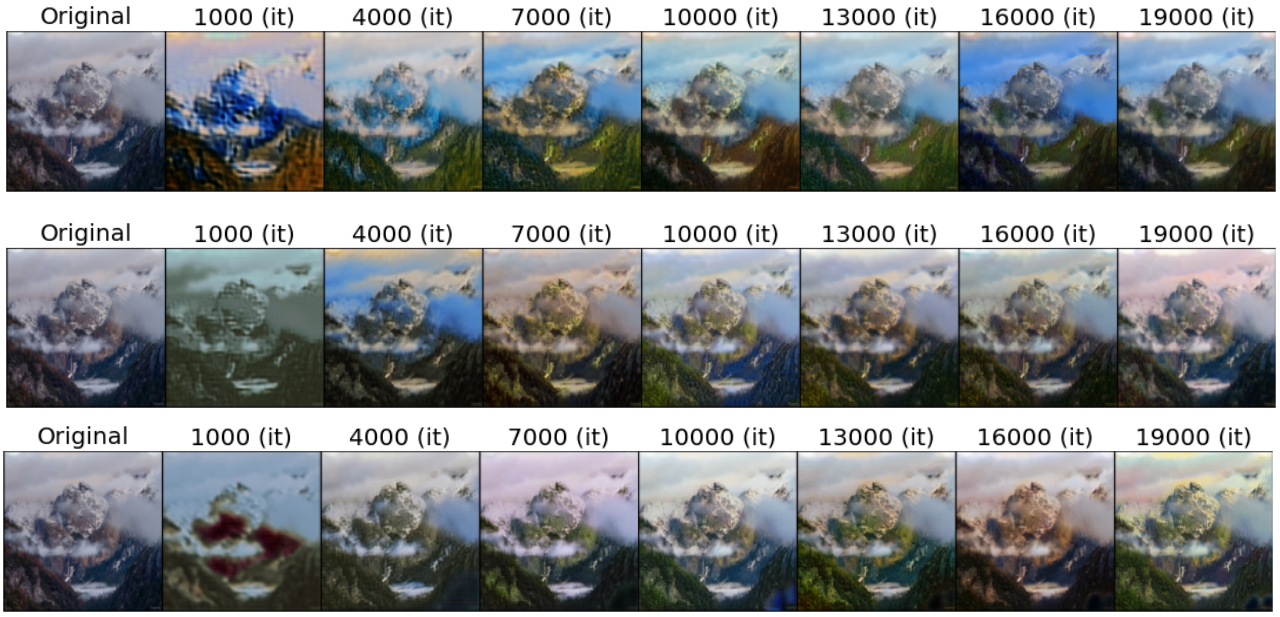

Figure 4. Model 1, 2, and 3 in respective order

The second part of generator loss is known as cycle consistency loss. This loss equation feeds an image into the first generator that maps $A \rightarrow B$ and then takes that output, runs it back through the generator that maps $B \rightarrow A$, and compares the output to the original image. If G is the generator in one direction, and F is the generator in the other direction, cycle consistency is given by:

$$\mathcal{L}_{cyc}(G, F) = E_{x \sim p_{data}(x)} \left[\|F(G(x)) - x\|_1 \right]$$

$$+ E_{y \sim p_{data}(y)} \left[\|G(F(y)) - y\|_1 \right]$$

Summing those two loss equations gives the total generator loss. To get the data for the graph above, checkpoint weights were saved every 1,000 iterations. Then, after training was done, we put 50 test images through the GAN and averaged to get the loss for every weight. We only used the original data, not the k-means images even if the model had been trained on quantized images for that weight iteration.

For qualitative results, we used used the saved weights from all three models and applied them to the same ten test images. For all three models, we used photos of Yosemite in the winter as inputs to the generator that is supposed to perform a style transfer and output a summer version of the same image. In Figure 3 and Figure 4 we have provided two examples.

7. Discussion

First, on the quantitative results. We can see that for the early weights, between iterations 1000 and 3000, the two k-means models perform far worse than the control. This is to be expected, because they have had severely limited exposure to color at that point, and the input images they are being tested on are not close to the distribution they have been train on at that time. We then see that very quickly, model 2 in particular records less loss than the control. This continues until the end of training when all three models start to converge. While this is an encouraging sign about the validity of the k-means quantizing approach, generator loss is often not a great indicator of image quality. The purpose of an image generation is to produce something realistic, and sometimes that is not always inline with the mathematically lowest loss.

The qualitative outputs, images that have had the style transfer applied to them, help us grasp the ways in which the k-means algorithm is improving training for CycleGAN. The two k-means algorithms both do a better job at using naturally occurring colors that the original model. In Figure 3, we can see that model 1 transforms the sky into an unrealistic orange, something that does not occur in either model 2 or 3. In Figure 4 we can see the original model generates an unnatural blue that the k-means models avoid. Qualitatively, model 2 produces the best looking images.

8. Conclusions and Future Work

From the results, we can see that pre-processing data with k-means clustering in the color space, and quantizing images, can help the CycleGAN model train. Specifically, it helps the model avoid creating images with unnatural colors that immediately look fake to the viewer. Compared to the control, our proposed approach yields more realistic generated images in fewer iterations of training. Therefore, our approach speeds up training time for CycleGAN on this image translation task. We are unsure why model 2 produces slightly better outputs than model 3, but assume it is because $K = 5$ is to many colors to start at to reap the full rewards of the algorithm.

The Inception Score is a widely used metric to quantify the quality of images generated by GANs. The metric captures both image quality (does the image look realistic, as a human might perceive it) and image diversity (the extent to which the model generates a large range of different images). Calculating the inception score requires employing a pre-trained Inception v3 model to classify each image generated by the generator. Then, KL divergence is used to measure how much the conditional probability distribution differs from the marginal probability distribution. Calculating the Inception Score is beyond the scope of our current project, but we hope to find a better way to measure generator performance in the next iteration of our project.

Another future improvement to this project would be a more systematic selection of K values for training. While we had rationale to guide our K-value selection, with more time we could have employed an iterative approach, and hopefully selected values that increased model performance even more. One interesting approach would be to oscillate each epoch between quantized and un-quantized images, and see if that helps model performance.

One last improvment is data augmentation. Even though data augmentations is one application of GANs, we could still apply data augmentation to increase the dataset for GANs, by applying image transformations (rotations, translations, shearing). Especially when collecting and pairing images is time consuming (e.g. for the home renovation and fireplaces datasets), performing simple data augmentation techniques can help the GANs achieve better performance with fewer epochs.

Contributions

We both worked together on data collection, implementation of the CycleGAN model and training. We were advised by

Sharon Zhou and are memebers of the GANs for Good study group in Stanford Machine Learning Group under Professor Andrew Ng, where we learned to train CycleGAN. This project is not, however, for that group or any other class. For the CycleGAN model, portions of the code were already written for the upcoming GANs Coursera course, and we read the paper and filled in the missing elements. Our contributions are clearly labeled in the code. All of the k-means data augmentation and changes to the model to be able to switch datasets between epochs are our own contributions.

References

Brejcha, J. and Čadík, M. Geopose3k: Mountain landscape dataset for camera pose estimation in outdoor environments. *Image and Vision Computing*, 66:1–14, 2017.

Efros, A. A. and Freeman, W. T. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 341–346, 2001.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

Kasuga, H., Yamamoto, H., and Okamoto, M. Color quantization using the fast k-means algorithm. *Systems and Computers in Japan*, 31(8):33–40, 2000.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.