Generative Styled Network for Creating Computer Graphics

Stephanie Dong
Stanford University
sxdong11@stanford.edu

Jeff Chen
Stanford University
jcl@stanford.edu

Nick Guo
Stanford University
nickguo@stanford.edu

Abstract

The idea that a computer can generate graphics in a desired style directly from a text captions is very powerful and its applications are incredibly useful. In this paper, we present a Generative Styled Network (GSN), which is capable of taking as input a text caption and some style images, and proceed to produce an image on a white background that matches both the text description i the style desired. We present baseline results, which are incomprehensible images and show our improvements: first with single category models, then with multi category models, and finally with the combined GSN.

1. Introduction

Creating original graphics for video games and software products in general is a costly and time consuming process. For example, graphics for video games such as items and buildings are repetitive in nature. Consider an online social game such as Farmville, which regularly introduces new items into the game. A graphic artist is typically given text descriptions of the new image assets, then he or she looks up what each item may look like, and finally draws each item in the game's artistic style.

This process is time consuming as it takes a long time for the artist to create each image, and often the artist is required to make several variations of the desired object for selection by management.

The input to this process is simply a text caption and example images of other graphics in the product, and the output is a graphic that matches the caption in the style of other images. Unlike other generative technologies, generating computer graphics has three key additional requirements 1) The graphic must be isolated on a white background 2) The graphic must adhere to the input caption 3) The image must fit with the style of other graphics.

We present a Generative Styled Network (GSN), which combines the architecture of a Generative Adversarial Network along with a Style Discriminator that makes it possible to generate such an image. The architecture consists of a Generative Adversarial Network with an Integrated Style Discriminator, joined together by simultaneously optimizing for the Generator loss, the Discriminator loss and the style transfer loss.

Generating original graphics from text is a very difficult technical challenge, because the output is highly modal since there are many arrangements of pixels that satisfy the text requirement. This makes training difficult as it is difficult to evaluate whether the generated image is acceptable. Generative Adversarial Networks have made remarkable strides in this area recently due to the simultaneous training of an image Generator and a critic in a 2 player min-max game [3].

Isolated style transfer, where the style is only transferred onto the object is motivated by the requirement that the graphic must appear on a white background, is also challenging since is difficult for the style transfer network to understand the location of the subject. Inspired by recent work on style transfer using CycleGan, where multiple discriminators and multiple Generators work together to isolate the area of the image where style transfer should happen, we add an additional style Discriminator directly to the Generator [16].

Our approach is to first isolate the individual challenges and then combine working pieces into a single stage network. We make use of the MS-COCO dataset for image generation as well as a proprietary dataset for game styles. Aided by the quantitative evaluation of generated images using the Inception score, we demonstrate remarkably good results with the combined architecture.

2. Related Work

There are two major components to our work: generating images from captions, and stylizing the generated images. The existing literature is rich with techniques for both domains, but we notably did not find previous work that combines text-to-image generation and style transfer.

Recent advances in Generative Adversarial Nets [3] show that one can generate images given text input. Mansimov *et al*. demonstrate that it is possible to learn visual representations from text by training a GAN network that uses bidirectional RNNs to learn the representations from text fed into variational auto-encoders [8]. Most notably, the experimental results reported by Mansimov *et al.* show that decent representations are learned, but the images are not realistic enough *(i.e. given an image generated by their GAN, a human would have a hard time discerning what class that image belongs to, but given both the image and the class, a human would be able to see the resemblance).*

Later advancements by Reed *et al.* show that there is potential to lightly vary the style of the image (background, placement, presence of other objects, pose) given sufficient context rich captions in the training set [10]. Given the stronger performance by the work of Reed *et al.*, we opt to base our architecture off of theirs.

Our work has a heavy experimental focus, so we need a way of quantitatively measuring the performance of one model over another rather than simply examining the outputted images by hand. Thus, we use the Inception score as a primary evaluation metric for our unstylized GAN outputs [12]. The Inception score is essentially computed by running a separately trained image classifier on the images generated by the GAN to evaluate the prediction confidence. As shown by Salimans *et al.*, the Inception score is highly correlated with how a human would judge the quality, or ”realness”, of an image.

In addition, using GANs [3] to generate artwork and image assets via text-to-image synthesis is fast and inexpensive, while observing all copyright laws since the images would be original creations.

For our style transfer baseline, we leverage the work of Gatys *et al.* as a benchmark for our post-processing step. We note that post processing is a good way to make headway on our problem, but this benchmark technique is essentially performing an optimization minimization to transfer textures and colors [1], so it cannot take advantage of deeper representations of subject specific assets from the graphic images. Thus, we explore the efficacy of integrating style transfer into our GAN model by incorporating an additional discriminator similar to the CycleGAN proposed by Zhu *et al.* [16]. By incorporating the notion of style when training our GAN, we aim to see if it is able to generate higher quality stylized images than the pipeline that stylizes as a post processing step.

3. Dataset and Features

3.1. MS-COCO

For content images and corresponding captions, we make use of the MS-COCO [7] 2017 dataset, which includes 118K training images and 5K validation images. These images are divided into 12 super-categories of [outdoor, food, indoor, appliance, sports, person, animal, vehicle, furniture, accessory, electronic, kitchen] and 80 fine-grain categories. MS-COCO also provides 5 annotations per image in addition to object segmentation masks.

We scanned over the available categories and their associated captions, focusing our model training efforts on three classes: elephants, laptops, and trains. There are 2143, 3524, and 3588 available training images respectively, and we chose these classes because they each have distinct features that should make them distinguishable. We only included image-caption pairs in our training set if at least one of the captions associated with an image contained the appropriate class name. We also preprocess the images by segmenting and cropping to focus onto the subject, the process for which is described in Experiments. We demonstrate that preprocessing is a necessary step that allows us to achieve better results.

3.2. Style Transfer Dataset

To perform style transfer, we train our GSN model using our in-house datasets of 3,000 stylized asset images from the Mafia LIVE! game [14] [2], and 2,000 asset images from the Undead LIVE! game [4].

4. Methods

4.1. Metrics

Machine generated images are notoriously difficult to measure. Salimans *et al.* introduce the Inception score as an evaluation metric for the quality of images outputted by GANs that correlates well with human judgment [12]. The score works by applying the Inception model to generated images in order calculate the conditional probabilities $p(y|\mathbf{x})$ assigned to each class y for the generated image $\mathbf{x}$. Ideally, each image generated by a GAN should be heavily concentrated in one class so that $p(y|\mathbf{x})$ has low entropy. At the same time, the variety of generated images should correspond to multiple classes so that $p(y)$ has relatively high entropy; otherwise, the model suffers from mode collapse where there is very little variance in the images outputted. The Inception score is then captured by the KL divergence formula given below:

$$\exp(\mathbb{E}_x \mathbf{KL}(p(y|\mathbf{x})||p(y))) \tag{1}$$

As we experiment with many different models and hyperparameters, we rely heavily on the Inception score to judge the quality of our network. Due to training efficiency constraints, we also present the results from models trained solely on one class of images, and in these cases we omit the multi-class component of the Inception score. We found it important to produce viable models for the single class instance prior to moving onto models for multiple classes, and for those we use the complete inception score.

4.2. Two Stage Generative Adversarial Learning and Style Transfer

Since we have to solve two different challenges 1) Generating an image from a caption, and 2) Fitting that image into a particular style, one intuitive approach is to take on these challenges separately and then chain them together into one system. The benefits of this approach is that the individual subsystems are smaller, which makes the challenges easier to solve and the models easier to train. However, a drawback of this approach is that the combined system will exhibit cascading errors. (e.g. if the generated image is far off, there is no chance the styled image would be acceptable, and vice versa).

We elect to start with a staged system to get results more quickly and then move to a more sophisticated integrated system.

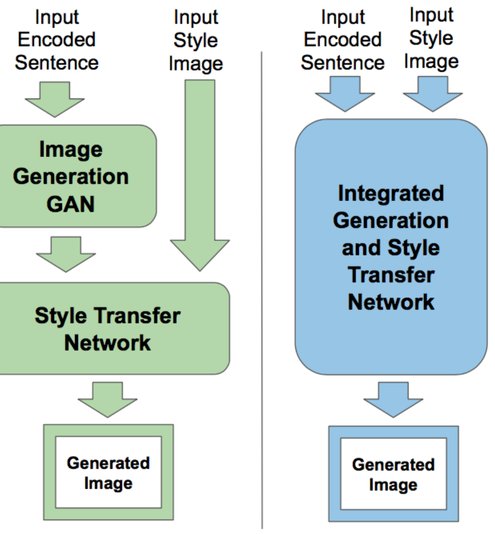


Figure 1. We face the dual challenge of generating an image from an encoded sentence as well as styling that image to the a desired existing style (such as that of a game or software product). We start initially with a staged approach (left), where we first generate the image and then apply styling. This is a good first step as the individual systems can be tuned separately. Then we move to an integrated architecture (right) that includes both image generation as well as style transfer.

4.2.1 Stage 1: Text to Image Generative Adversarial Learning

The first component of the staged system must be able to generate an image from a caption, such as "A long red and black train." To accomplish this, we first encode the caption into something a computer can understand, and then generate the image.

First presented by Kiros *et al.*, Skip-Thought Vectors is a way to encode a sentence into a vector [6]. The model optimizes for the objective shown in Equation 2, when given a tuple of sentences (s_{i-1}, s_i, s_{i+1}).

$$\sum_t \log P(w_{i+1}^t | w_{i+1}^t, h_i) + \sum_t \log P(w_{i-1}^t | w_{i-1}^{<t}, h_i) \tag{2}$$

Effectively, the model tries to tune the weights of network in such a way that maximizes the probability of its neighbors. This is very similar to the idea of SkipGrams in Word2Vec. Since we are are trying to capture the meaning of the entire sentence, we picked Skip-Thoughts vector as our word encoder, and obtained a pre-trained model on the BookCorpus dataset [17].

To generate the actual image, we need a way to transform the text encoding into an image. We propose the use of a Generative Adversarial Network(GAN) to for this task, which have demonstrated much better ability to generate images than historical PixelRNN/CNN [15].

A GAN uses two networks where a Generator network competes with an adversary, the Discriminator, with the min-max overall objective show in Equation (3).

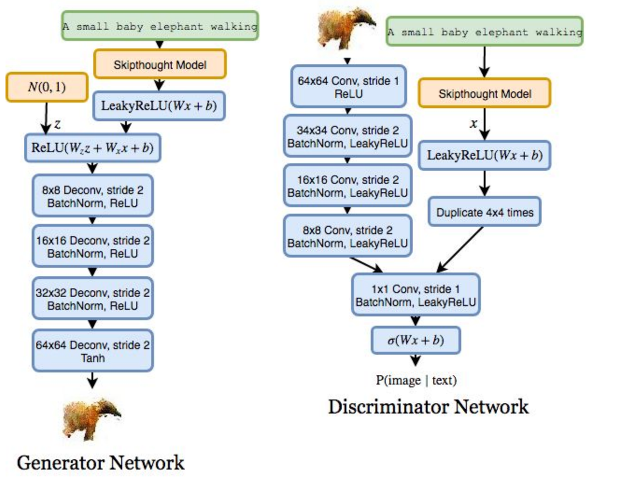
$$\min_G \max_D V(G,D) = \mathbb{E}_{x\sim p_{data(x)}}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{3}$$

This network acts like a 2 player game, where each player tries to best the other. The Generator is trying to produce images that fool the Discriminator, which will decrease its loss. Meanwhile, the Discriminator is trying to improve its ability to distinguish real images and fake images, which will decrease its loss.

We apply a GAN to our challenge and pass the encoded caption to the Generator along with a 100 dimension noise vector. The Generator first reduces the dimensions of the caption embedding to 256 with a linear layer, then concatenates it with a 100 dimensional noise vector, then runs the output through another linear layer to yield the desired number of features. The features are then run through 4 de-convolutional layers with stride 2 that upsample the input on every layer. The Discriminator takes the generated image, passes it through 4 convolution layers with stride 2, with the result in as a 4x4 image, and then adds the encoded text vector spatially before finally performing a 1x1 convolution with rectification and a 4x4 convolution to generate the final Discriminator score.

We start our investigation with the Text to Image Synthesis architecture proposed by Reed *et al.* [10], and an implementation by Paarth Neekhara [9]. The objective function used for the Discriminator is to minimize $\log(G(D(x)))$ since $\log(1 - G(D(x)))$ saturates early on because the discriminator would be perfect in detecting bad images.

While this architecture works well on the Oxford-102 Flowers dataset, several critical modifications are required for task of generating styled and isolated graphics.


A small baby elephant walking
Skipthought Model
N(0, 1)
LeakyReLU(Wx + b)
z
ReLU(W_z z + W_x x + b)
8x8 Deconv, stride 2 BatchNorm, ReLU
16x16 Deconv, stride 2 BatchNorm, ReLU
32x32 Deconv, stride 2 BatchNorm, ReLU
64x64 Deconv, stride 2 Tanh
Generator Network
A small baby elephant walking
64x64 Conv, stride 1 ReLU
Skipthought Model
34x34 Conv, stride 2 BatchNorm, LeakyReLU
x
LeakyReLU(Wx + b)
16x16 Conv, stride 2 BatchNorm, LeakyReLU
Duplicate 4x4 times
8x8 Conv, stride 2 BatchNorm, LeakyReLU
1x1 Conv, stride 1 BatchNorm, LeakyReLU
σ(Wx + b)
P(image | text)
Discriminator Network
Figure 2. Architecture for the Generative Adversarial Network[10]

First, the image we generate should isolate the object without any background. Second, the we need to demonstrate that it is possible to generate more object categories than just flowers. Third, since it's difficult to find perfectly manicured datasets such as birds and flowers, we would like to be able to train on the MS-COCO dataset, which may have more than one object in the image. And finally, we require the final generated image to be in a particular artistic style.

4.2.2 Stage 2: Style Transfer

We evaluate the style transfer architecture by Gatys *et al.* as a baseline for our needs[1]. In the context of the original problem, the assets from which we want to transfer style are typically based on a white background, so we expect that the baseline style transfer model will not perform well when overpowered by the white pixels.

4.3. Generative Styled Network

Motivated by works in unsupervised image generation networks, and CycleGAN [16], we propose that a deep convolutional neural network will be better at representing the style of an image or a distribution of images compared simply employing the Gram Matrix, which was proposed by Gatys *et al.* [1].

To this end, we devised our own multi-Discriminator scheme to learn both a distribution over content images supported by textual descriptions and also learn the distribution over style images. Together, the 2 Discriminators jointly teach the generator to produce images in the given style and corresponding to the input textual descriptions.

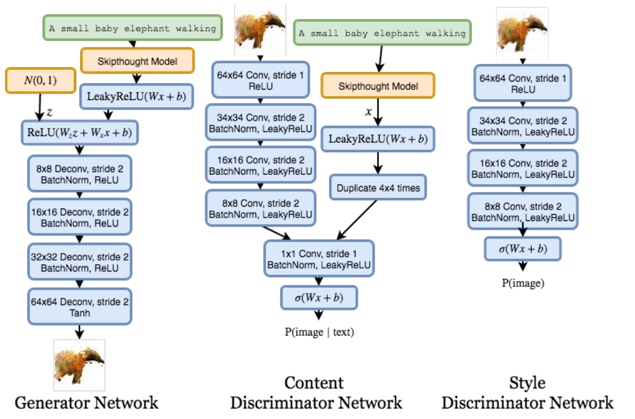
We add a Style Discriminator to the GAN model with the role of discerning if a given images belongs in the style distribution. This new Discriminator is used to augment the Generator loss to encourage the Generator also generate images in that mimic the style distribution.

Formally we define the Style Discriminator S, trained on a set of images s_i drawn from the distribution of images of the desired style $\mathcal{S}$, that given an image x, produces the percentage probability that x was drawn from $\mathcal{S}$.

The resulting two discriminator GAN has an min-max-max overall objective show in Equation (4).

$$\min_G \max_S \max_D V(G,S,D) = \mathbb{E}_{x\sim p_{data(x)}}[\log D(x)]+ \mathbb{E}_{s\sim p_{style(s)}}[\log S(s)]+ \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))+\log(1-S(G(z)))] \tag{4}$$

Figure 3 depicts the full model architecture for our Integrated Style Transfer GAN.


A small baby elephant walking
Skipthought Model
N(0, 1)
LeakyReLU(Wx + b)
z
ReLU(W_z z + W_x x + b)
8x8 Deconv, stride 2 BatchNorm, ReLU
16x16 Deconv, stride 2 BatchNorm, ReLU
32x32 Deconv, stride 2 BatchNorm, ReLU
64x64 Deconv, stride 2 Tanh
Generator Network
64x64 Conv, stride 1 ReLU
34x34 Conv, stride 2 BatchNorm, LeakyReLU
16x16 Conv, stride 2 BatchNorm, LeakyReLU
8x8 Conv, stride 2 BatchNorm, LeakyReLU
A small baby elephant walking
Skipthought Model
x
LeakyReLU(Wx + b)
Duplicate 4x4 times
1x1 Conv, stride 1 BatchNorm, LeakyReLU
σ(Wx + b)
P(image | text)
Content Discriminator Network
64x64 Conv, stride 1 ReLU
34x34 Conv, stride 2 BatchNorm, LeakyReLU
16x16 Conv, stride 2 BatchNorm, LeakyReLU
8x8 Conv, stride 2 BatchNorm, LeakyReLU
σ(Wx + b)
P(image)
Style Discriminator Network
Figure 3. Architecture of Integrated Style Transfer GAN

5. Experiments and Results

5.1. Baselines

We evaluated baselines on both multi-category and single category image generation. The multi-category baseline includes 10 categories: car, airplane, boat, bus, horse, elephant, motorcycle, tv, refrigerator, bear. We evaluated single category baselines on train, elephant, and laptop. Our selection criteria for categories are 1) the object should naturally be large, since it would be more likely to be the focus point of the image, and 2) the number of training images should be greater than the median across all categories.

Our initial baseline evaluation produced Inception scores of 2.48, 3.75, 2.2 for elephant, laptop, and train single category models. The multi category Inception baseline score is 2.49. Shown in Table 1, the generated images are incomprehensible. We see elephant as a large brown in front of a green background, laptop is a screen on top of black, and trains are non-descript black regions on top of what may be sky.

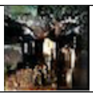
Caption	**Generated Image**
Multi Category: A large elephant	
A small elephant walking across a dirt field.	
A black laptop with a blank screen sitting on a desk.	
A yellow and black striped train next to sidewalk.	

Table 1. Baseline results. The first row is trained with 10 categories, and we quickly realized that we needed to simply the challenge before training on a multi category model capable of generating images for more than one category. The bottom three rows show baseline results for models trained on single categories elephant, laptop, and train. Overall, baseline images are incomprehensible.

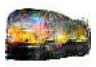
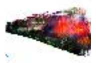
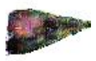
Captions for Segmented and Cropped Images (Train Category)	**Generated Image**
A yellow and black stripped train next to sidewalk.	
A red train is docked at the station.	
A large long train.	

Table 2. Segmenting and then cropping images significantly improved generated image quality. Importantly, our model is able to capture information in the caption. Shown in this table, colors and length are clearly captured. However, information about the surroundings is no longer captured, such as ”docked at the station.” This is expected since we segment out the background.

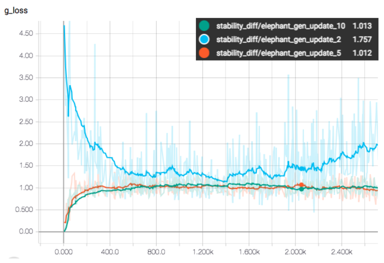


Figure 4. Generator Loss with 2, 5, 10 Generator updates per Discriminator update.

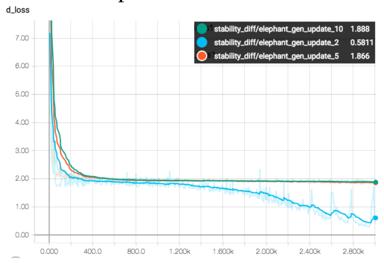


Figure 5. Discriminator Loss with 2,5, 10 Generator updates per Discriminator update.

5.2. Image Segmentation and Cropping

Since even the single category baseline generated images very far from human recognizable objects, we take the approach of improving single category image generation before moving to multi category image generation. We suspect this is due the MS-COCO dataset containing many object classes in each image, which causes our GAN model to essentially learn to optimize against noise.

We hypothesize that segmenting the images to mask out non-elephant pixels will reduce the noise fed into the network and will make it easier for our model to learn.

MS-COCO provides segmentation information for objects in its images, based on category, which allows us to segment the image and isolate the subject. For example, we used the MS-COCO class annotations for trains to mask out non-train pixels and then crop the resulting image to the smallest centered square bounding box on the train. Further, we reject images that constitute less than 7% of the image by area, since those objects are not likely the focus of the caption.

This was highly successful and we were able to increase our Inception score substantially, especially for the train category (2.2 to 5.12) as shown in Table 6. These results are also much better based on human judgment as we can now clearly make out a train. Example generated images are shown in Table 2.

Critically, we observe that details in the caption are captured such as color, as ”A yellow and black striped train next to a sidewalk” actually shows a yellow and black train and ”A red train is docked at the station” shows a red train.

5.3. Hyperparameter Tuning

Training a GAN is very tricky and a lot depends on the ratio of updates to the Discriminator versus the Generator. For example, updating the Discriminator too frequently would lead to the Generator not being to learn quickly enough to catch up, which would lead to sub-par images.

Ideally we'd want the loss of both the Discriminator and the Generator to stabilize, which indicates that just as the Generator is able to fool the Discriminator, the Discriminator learns to discriminate the images - this is the ideal scenario that leads to high quality generation.

Figures 5 shows the effect of changing the update ratio between the Generator and the Discriminator. We notice that as the number of epochs increase, image quality gets better, eventually reaching a maximum and then starts to decrease. We also see that in our particular model, using a update ratio of training the Discriminator for 2 epochs before updating the Generator is ideal. Due to limited compute, we did not further tune other hyperparameters: Optimizer (Adam) and Learning Rate (0.0002) since our initial selection, presented by Reed *et al*. produced reasonable images.

5.4. 5-Layer / 6-Layer / VGG Architecture

Recent improvements in visual recognition depend on deeper networks, we additionally hypothesize that adding additional layers to our network will lead to more realistic generated images. Based on a visual analysis of our segmentation results, we notice that refined features are missing from the generated images. To this end, we chose to add additional layers to our Generator network, with the layers mirrored in the Discriminator.

The first architecture change we made was adding an extra deconvolutional layer at the last stage of the Generator, when the image is already at the full size of 64 pixels. This resulted in an improvement in the Inception score from 3.27 to 3.64 for the elephant category as shown in Table 6.

We also attempted adding an additional deconvolutional layer when the image is one half the final size, at 32 pixels. Though this performed poorly and we were not able to improve the generated image quality.

Additionally, we had the intuition that since the VGG network performed well for image recognition it may also improve performance the of our GAN [13]. The actual VGG network uses repeated standard modules of convolution-convolution-max_pool. Since the generator essentially uses the back-propagation pipeline of a classifier, we propose 4 repeated layers of deconv_stride_1 - deconv_stride_2. We directly upsample the image with a deconv_stride_2 layer instead of using an unmax_pool layer because this makes training more stable. Though this model did not yield better performance with an Inception score of 3.27.

5.5. Transfer Learning

Orthogonal to our experiments in architecture modification, we also experimented with transfer learning to achieve more realistic generated images.

Reed *et al*. [10] generates realistic flower images based on Oxford-102 Flowers dataset, and [9] provides a model pretrained for 600 epochs on this Flowers dataset.

We hypothesize the lower level weights and features that inform realistic flower discrimination and generation will also apply to other classes of subjects. To this end, we initialized the first and second convolution and BatchNorm layers in our basic GAN model with the pretrained flowers model. Additionally, we initialize the first and second BatchNorm layers, the embedding layer, the first linear layer, and the first convolution layer of the generator with the pretrained flowers model.

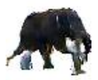
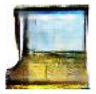
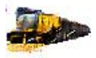
Caption	Generated Image
A small elephant walking across a dirt field.	
A black laptop with a blank screen sitting on a desk.	
A yellow and black striped train next to sidewalk.	

Table 3. Sample results generated with pretrained weight initialization trained on single category segmented images.

Based on human judgment, the resulting generated imaged produced by the GAN initialized using pretrained weights are more realistic in general, but are much less colorful and represents a less diverse number of subject orientation and poses. Likewise, the images generated can present a higher mismatch with the captions as compared to the model train without initializing with pretrained weights. Sample of results in Table 3.

5.6. Generative Styled Network

Our baseline style transfer applied the architecture by Gatys *et al*. [1] directly to the generated images. This technique has a limitation of only being able to reference a single style image, and used the Gram Matrix of style image activations to capture the style. As shown in Table 4, applying the Gatys *et al*. style transfer directly fails to transfer the style meaningfully to the generated image. Furthermore, applying the technique naively greatly distorts the original generated image and makes them unrecognizable. Our segmented style transfer similarly was not successful in applying the style to the generated images.

In contrast, our integrated style transfer method had a much higher success in both generating realistic images that correspond to the textual captions, as well as, applying the style of the references images to the generated image.

We see the generate image for a train skewing toward a darker color scheme, and the laptop gets strong blue highlights. These are qualities not present in the training set and not present in generated images without addition of the Style Discriminator.

5.7. Multi Category Performance

After achieving satisfactory results with single category models, we return our attention to models with multi category generation capability. Our improved model gener-

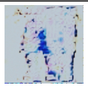
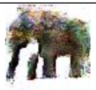
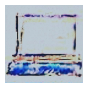
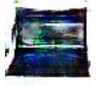
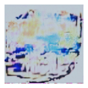
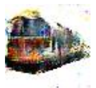
Integrated GAN and Style Transfer Network	Baseline Style	Integrated Style
Style Assets From Mafia Game		
A small elephant.		
A laptop computer is displaying a screen with words.		
A green train is coming down the tracks.		

Table 4. Baseline Style Transfer and Integrated Style Transfer trained for single category models.

ates an Inception score of 7.15, which is a significant improvement over the baseline socre of 2.49. Based on human judgement, elephants, laptops, and trains are very easy to recognize. Example images are shown in Table 7.

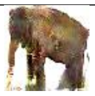
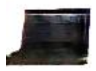
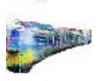
Multi Category Model Caption	Generated Image
A picture of an elephant standing in some brush.	
A black laptop with a blank screen sitting on a desk.	
A green train is coming down the tracks.	

Table 5. Following the success of being able to generate images with a single category model. We return to the challenge of training a single model to produce images across many categories. Shown here, the model shows little confusion between captions about elephants, laptops, and trains, showing our encoding of sentences is able to inform the model of the subject as well as image details such as color.

We also see that additional information in the caption is also being recognized by the Generator, such as information about the color of the object.

6. Discussion

6.1. Overall Performance

Our best performing Generative Styled Network is able to generate images from a single category and simultaneously style the image based on a set of style images (presumably other graphics from the same target application). Shown in Table 4, generated images are instantly recognizable and resemble features of our style dataset.

To achieve this result, we had to first make dramatic improvements on single category image generation, summarized in Table 6. We increased our primary metric, the Inception Score from 2.48 to 3.63 for the elephant category. Qualitatively, the images are significantly better and we present single category results in Table **??**.

Single Category Improvements	Elephant Score	Laptop Score	Train Score
Integrated GAN and Style Transfer	3.33	4.87	3.94
5 Layer Network	**3.63**	-	-
VGG Architecture	3.27	-	-
6 Layer Network	3.50	-	-
Transfer Learning	2.84	-	-
Segmented and Cropped Images	3.55	4.4	5.12
Segmented Images	3.54	2.93	4.4
Baseline	2.48	3.75	2.2

Table 6. Inception Scores for Single Category Image Generation. Our experiments in performing segmentation and cropping on the training images had a significant improvement. Further, our architecture change to increase the network to 5 Layers further improved performance. For generating elephants, our model improved from a baseline Inception score of 2.48 to 3.63. Integrated GAN and Style Transfer network scores are provided here for reference - we expected the degrade in Inception Score as game styles are applied. Not all experiments were run, denoted by '-', since each model takes 2 days to train on a K80 GPU.

Following the success of our Generative Styled Network on a single model, we proceed to apply the model in the multi category case. Interestingly, our improvements in the single category case did not fully transfer. Show in Table 8, we achieved the highest Inception score at 7.15 with just the Segmented and Cropped images. Particularly, our GSN preformed merely achieved an Inception score of 1.86. We hypothesize this is due to the increase of batch size from 256 to 1024 - due to the larger network and the combined number of training images, we used a larger batch size to make training time more reasonable. Though this may have had an adverse effect since in the beginning of image generation, we would prefer the generator and Discrimintor to be more random instead of more stable (large batch sizes). More stability with larger batch sizes may quickly force the system into an undesirable saddle point, from which it cannot escape.

6.2. Failure Analysis

We performed analysis on our best performing multi category model to gain more insight on what is driving model

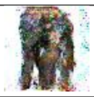
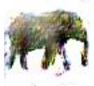
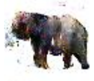
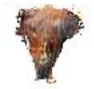
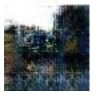
Improvements for Single Category model: A small elephant walking across a dirt field.	**Generated Image**
Integrated GAN and Style Transfer	
5 Layer Network	
Segmented and Cropped	
Segmented	
Baseline	

Table 7. Improvement over the baseline for Single Category image generation. Segmented Images provide the white background we are looking for. Cropping images provides the definition of the elephant. The 5th layer gives the finer detail and a sense of motion. Integrated style transfer provides a shift in color and tone closer to the desired style provided by the style transfer assets

Multi Category Improvements	**Inception Score**
3-Category Segmented and Cropped	7.15
3-Category Segmented and Cropped 5-Layer	6.03
3-Category Segmented and Cropped with Transfer Learning	5.67
Multi-Category Baseline	2.49
3-Category Integrated GAN and Style Transfer	1.86

Table 8. Inception Scores for Multi-Category Image Generation. Surprisingly, not all of our improvements carried over to the multi category application, especially the integrated Generative Styled Network.

behavior. Shown in Table 9, we see that the shape of the elephant conformed to the learned shape of “red train” rather than the color.

Further, we note that segmentation may lead to misunderstanding by our model. Shown in the bottom of Table 9, the word ‘green’ causes generated assets to have an enlarged shape, similar to an elephant. We identify that many elephant captions have the word ’green,’ yet the backgrounds containing the green foliage have been segmented out. This causes the network to associate ‘green’ with elephant rather than with the color because it again has insufficient information on what ‘green’ should be. We believe that cropping down to the elephant instead of segmenting and cropping provides a potential solution at the expense of lowering the generated image quality. As is typical with deep networks, a larger and more sanitized dataset would improve the quality of our model.

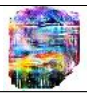
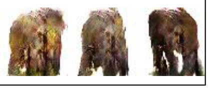
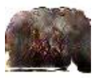
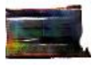
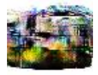
Failure Analysis Examples		
A red train car that has graffiti	a red laptop	a red elephant.
		
A green elephant.	A green laptop.	A green train.
		

Table 9. Two shortcomings in our best model trained on elephants, laptops, and trains. Above, the network incorrectly associates ‘red’ with ‘train’ because there were red trains in the training set, so a ‘red elephant’ shows an elephant in the shape of a train. Below, the color ‘green’ is associated with elephants precisely because green is missing from the training set (e.g. a training caption was ”Two elephants on a field of green of green grass.”, but the green background that should be present was cropped out.)

7. Conclusion

We developed a Generative Styled Network (GSN) for generating styled computer graphics on a white background directly from a text caption. The network simultaneously generates an image and sets it to the correct style. To the best of our knowledge, we are the first to present a combined architecture for image generation and style transfer and also the first to generate isolated objects on a white background. Through segmentation, cropping, and architecture changes, we dramatically improved the performance of our image Generator from an Inception score of 1.86 to 7.15 for multi category image generation. We improved the Inception score for the elephant category from 2.48 to 3.63. Qualitatively, the difference of the generated images to a human is staggering. Our baseline is incomprehensible, while our final results are easily distinguishable as elephants, trains, and laptops with specific features (such as color, length, size) as described by the text caption.

In future work, we will improve the performance of GSN for generating and styling images in multi-category models, improve the fidelity of the output image, as well as increase the output image size.

8. Contributions and Acknowledgements

Stephanie worked on implementation of Transfer Learning and the Generative Styled Network, Jeff implemented the data loader to work with MS-COCO, architecture improvements, hyperparameter search, and processing pipeline. Nick implemented the Style Transfer baseline, segmentation and cropping. All contributed to the writing of the report.

We would like to thank Paarth Neekhara for sharing his implementation [9] of the DC-GAN described by Reed *et al*. [10], which served as a starting point for our model. We'd also like to thank Ryan Kiros for sharing a pre-trained Skip-Thoughts [6] model which we used to produce our sentence vectors [5]. We heavily modified Neekhara's work to support the various architecture changes that we made, and to support an additional style Discriminator resulting in our Generative Styled Network model. We also would like to thank Tim Salimans and OpenAI for open-sourcing their implementation of the Inception score [12], which we adapted into our pipeline to evaluate the models [11].

References

[1] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. pages 2414–2423, June 2016.

[2] GiantBomb. An iphone game in which you are the don of your own mafia. `https://www.giantbomb.com/mafia-live/3030-25160/`, 2009.

[3] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *ArXiv e-prints*, June 2014.

[4] M. N. Jeremy. Apps for less: Mafia live! and undead live. `https://www.imore.com/apps-undead-live-mafia-live-aqua-moto-racing`, 2009.

[5] R. Kiros. skip-thoughts. `https://github.com/ryankiros/skip-thoughts`, 2015.

[6] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler. Skip-thought vectors. *CoRR*, abs/1506.06726, 2015.

[7] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[8] E. Mansimov, E. Parisotto, L. J. Ba, and R. Salakhutdinov. Generating images from captions with attention. *CoRR*, abs/1511.02793, 2015.

[9] P. Neekhara. text-to-image. `https://github.com/paarthneekhara/text-to-image`, 2016.

[10] S. E. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. *CoRR*, abs/1605.05396, 2016.

[11] T. Salimans. improved-gan, inception score. `https://github.com/openai/improved-gan/tree/master/inception_score`, 2016.

[12] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016.

[13] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[14] UniqueApps. Mafia live iphone app review. `https://www.youtube.com/watch?v=kq6KKn0oCgE&t=6m50s`, 2009.

[15] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional image generation with pixelcnn decoders. *CoRR*, abs/1606.05328, 2016.

[16] J. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017.

[17] Y. Zhu, R. Kiros, R. S. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *CoRR*, abs/1506.06724, 2015.