draw2pix: Generative Adversarial Networks for Art School Rejects

Cedrick Argueta and Kevin Wang
Stanford University
450 Serra Mall, Stanford, CA 94305
cedrick@cs.stanford.edu
kwang98@stanford.edu

Abstract

We implement a version of CycleGAN that includes various regularization and stabilization techniques applied in other types of GAN models, and use it to perform sketch-to-photo image translation. Additionally, we experiment with a more advanced loss function for generators that aims to enforce weaker cycle consistency early in training to ensure that generators produce realistic images. We discuss the trade offs that come from using automatically generated edge maps vs. hand-drawn sketches, and demonstrate various techniques that alleviate a majority of the generalization issues. The completed model implies the ability of CycleGAN to generalize an edge-to-image training regime to a sketch-to-image testing regime. Finally, we compare our results against a baseline CoGAN model, through both quantitative and qualitative metrics.

1. Introduction

Image-to-image translation has seen several improvements in recent years due to generative adversarial networks, or GANs. We are investigating unpaired image-to-image translation with generative adversarial networks. Work done by Isola et al. in [8] demonstrates how we can use paired images in training to create an image-to-image mapping from one domain to another. An extension to the pix2pix model is found in CycleGAN [21], where the loss function for the generative network contains a term that enforces *cycle consistency* across mappings, i.e., that a mapping from one domain to another can be reversed to generate the same image. The specific image-to-image task demonstrated in our work is similar to this visualization by Christopher Hesse, which demonstrates examples of sketch-to-photo translations trained using the pix2pix model. We implement a version of CycleGAN that contains various stabilization techniques found in other GAN models, such as experience replay, label smoothing, and noisy labels. We use this version of CycleGAN to perform sketch-to-photo translation. The relaxation that unpaired training provides allows for easier creation of datasets and combinations of domains – we swap a domain and retrain rather than find image-to-image pairs for that specific translation task. We implement a version of CycleGAN that contains regularization in the real and fake label used by the discriminator, and use a modified cycle consistency loss that enforces feature-level cycle consistency in addition to pixel-level cycle consistency. In addition to these improvements upon the original CycleGAN, we experiment with using artificially generated edge maps for training. Artificially generated edge maps are easier to produce for a set of images, though require various preprocessing techniques to allow the model to generalize to hand-drawn sketches. Our project differs from [22] in that we aren't doing multi-modal image translation, only from one domain to another. The inputs to our model are either sketch images or real photos, which are processed by the learned generative networks and transformed into real photos and sketches, respectively. In particular, we work with images of trees, as an homage to Stanford University.

2. Related Work

Generative Adversarial Networks

The original GAN proposed by Goodfellow et al. [5] constitutes a network that learns an approximation of a distribution of data p_X. It does so by training two networks - a generator network and a discriminator network. These networks are trained adversarially, i.e. the discriminator's objective is to discern real images from generated images, and the generator's objective is to generate images that fool the discriminator.

Generative networks are not limited to simple distributions. Work done by Liu et al. [11] demonstrated the ability of generative networks to learn a joint distribution between variables, through the use of multiple generator and discriminator networks. This can be used to generate pairs $x_1 \sim p_{X_1}, x_2 \sim p_{X_2}$ of images that belong to the joint distribution just by using the marginal distributions.

Supervised Translation

When labels are given, i.e. explicit pairings between the two domains, supervised methods like those described by Isola et al. [8] are possible through the use of conditional GANs. The pix2pix model described in this work performs single-modal translation with multiple generator networks, one for each domain-to-domain pairing. The model has been effectively applied to problems such as generating photos from label maps, reconstructing objects from edge maps, and colorizing images.

Unsupervised Translation

Neural Style Transfer [9] is an alternate way to perform image-to-image translation, which synthesizes an image by combining the content of one image with the style of another image. It does so by matching the Gram matrix statistics of pretrained deep features. In the context of our problem, the content image represents the image which we would like to translate, and the style image represents the domain that we would like to translate it to. However, this method is only applicable to single sample transfers, not entire collections. It is possible to take an ensemble average over a style collection of Gram matrix statistics, but this has been shown to be less effective than other methods such as CycleGAN.

CycleGAN is an example of unsupervised image-to-image translation, where no labels are given to pair images from domain X and domain Y. Zhu et al. [21] go a step further by enforcing cycle consistency between pairs created, i.e. by ensuring that generated images can be run through another generator to create the paired image. CycleGAN effectively learns a mapping function $G : X \rightarrow Y$ for images in domains X and Y, along with its inverse function $F : Y \rightarrow X$. Cycle consistency then amounts to the condition $F(G(x)) \approx x$ and $G(F(y)) \approx y$.

The release of CycleGAN spurred many variants that also solve the unsupervised translation problem using cycle-consistency.

Bansal et al. [1] implements Recycle-GAN, which solves the problem of unsupervised translation of video data by combining CycleGAN with a Recurrent Neural Network (RNN) to implement spatiotemporal constraints on the data. Essentially, a stronger consistency between domains is built by insisting not only that individual frame data is cycle-consistent, but also that the temporal features of the video are as well.

Liu et al. [12] solves the problem with Coupled GAN, which assumes a shared latent space z between domains X and Y. This assumption automatically implies cycle-consistency since the latent space implicitly defines a mapping from $X \rightarrow Y$ and vice-versa. Variational Autoencoders (VAEs) are implemented on the latent space to generate fake image data. Weight-sharing is used on the corresponding encoder and generator matrices of the two domains. Coupled GAN has been shown to have increased performance over CycleGAN.

3. Data

We make use of several datasets to perform translation. Training is primarily done with an ImageNet synset [17] of palm trees. In our problem formulation, we perform training on photos of trees and automatically generated edge maps of trees, and test our model by transforming hand-drawn sketches of trees to photos of trees.

Selecting a tree species to perform image translation with was challenging. Preliminary tests with the edge detection algorithm on the Linden tree synset showed that a majority of the Linden tree images were not iconic images of Linden trees, but rather closeup images of leaves, pictures with trees in the background or as a small section of the image, or not Linden trees altogether. For these reasons, palm trees were chosen over more traditional tree synsets like Linden trees and oak trees, as palm trees have a tendency to be photographed against sky backgrounds. The palm tree synset contained 1605 images, split into a training set of 1578 and a testing set of 27 images.

Edges are automatically generated from these images through the use of bilateral filtering and the Canny edge detection algorithm [2]. We found early in experimentation that performing image translation with various types of background noise reduced image quality and made training more difficult with edges. Bilateral filtering was used to smooth images before applying Canny edge detection, reducing artifacts and spurious edges that added unwanted noise to the resulting images. We found that despite the Gaussian filter applied in the Canny algorithm, a bilateral filter applied beforehand further reduced noise in the produced edge maps. The algorithm calculates intensity gradients using filters convolved across the image, and then thins the edges and drops spurious edges through comparison with lower and upper bound parameters that we select. We select these parameters to be 235 and 250, respectively, giving a comparatively tight representation of the edges in an image. Because edges are generated from the original photos, we also have 1605 edge maps, split into a training set of 1578 and testing set of 27.

This approximation of sketches was used instead of hand-drawn sketches for several reasons. Hand-drawn sketches of trees from ImageNet were available from the Sketchy database [18]. Training generative networks requires a large amount of data, however, and there were comparatively fewer hand-drawn sketches from the Sketchy dataset than possible edge maps from tree synsets in ImageNet. A large part of our contribution includes the preprocessing of tree images on ImageNet to accurately represent

Figure 1. A palm tree and its corresponding edge map. Also pictured is an example of a tree from the SketchyDataset, not generated through an edge algorithm but instead drawn by hand.

sketches, allowing generalization of our model to sketch-to-photo synthesis from edges-to-photo synthesis. An example of a palm tree source image and the applied edge filtering algorithm is pictured in 1.

To test generalization to sketches, we also make use of the hand-drawn sketches. Hand-drawn sketches are taken from the Sketchy Dataset [18]. These sketches are representations of images from ImageNet, shown to subjects and then drawn by hand. The dataset is used to examine the generalization of our model to sketch-to-photo synthesis from a edge-to-photo training regime. The test dataset has 200 images of trees from ImageNet with 5 unique sketches each.

All images fed into the model are normalized in the same fashion: images are resized to 256 $\times$ 256 pixels, with the images normalized between to $[-1, 1]$ across all pixels. In order to combat overfitting with this small dataset, data augmentation is performed. Data augmentation is done on both edge and photo domains through random horizontal flipping over each epoch, and additionally perturbations in saturation, hue, and brightness are applied to the photo domain. There perturbations are only applied over a 10% range of the original value.

4. Methods

We use CycleGAN [21] for unpaired image-to-image translation. More precisely, we train networks to learn functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$. This allows us to take an image $x_1 \in X$ and generate its pair $y_1 \in Y$, and vice versa. We let the true distributions of the domains be p_X and p_Y. Just as in [5], we will use discriminators D_X and D_Y as adversarial networks to the generators, which we train to distinguish between elements in X and p_X and Y and p_Y, respectively. CycleGAN uses two losses per generator function. The first loss is adversarial, where the loss function for G and its discriminator D_Y is:

$$\begin{aligned}\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y\sim p_Y}[\log D_Y(y)] \\ +\mathbb{E}_{x\sim p_X}[\log(1 - D_Y(G(x)))]\end{aligned} \tag{1}$$

and the objective is $\min_G \max_{D_Y} \mathcal{L}_{GAN}$. We can apply this without loss of generality to F and its discriminator D_X. In this formulation of the loss function, the generator network minimizes the probability of being assigned a fake label for its generated images. The discriminator network maximizes the probability of assigning correct labels to real and fake images.

Because it is possible for a network to map an input image x to multiple images in Y, we use a cycle consistency loss to guarantee that there is a unique mapping. Specifically, we wish to enforce that for images x and y, $F(G(x)) \approx x$ (forward cycle consistency) and $G(F(y)) \approx y$ (backward cycle consistency). The loss used is:

$$\begin{aligned}\mathcal{L}_{cyc} = \mathbb{E}_{x\sim p_X}[\|F(G(x)) - x\|_1] \\ +\mathbb{E}_{y\sim p_Y}[\|G(F(y)) - y\|_1]\end{aligned} \tag{2}$$

This loss minimizes the L1 distance between a reconstructed image and its source image. This encourages the generators to learn inverse mappings.

Then the full loss function in the standard CycleGAN implementation is:

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) \\ +\mathcal{L}_{GAN}(F, D_X, Y, X) \\ +\lambda\mathcal{L}_{cyc}(G, F)\end{aligned} \tag{3}$$

where λ is a parameter that determines the importance of cyclic consistency loss vs. adversarial loss. Higher values of λ weight cycle consistency higher than image quality. Minimizing this loss through

$$G^*, F^* = \arg\min_{G,F} \max_{D_X, D_Y} \mathcal{L}(G, F, D_X, D_Y) \tag{4}$$

yields G^* and F^*, our optimal mapping functions.

A variety of tricks exist that aid in stabilizing GANs [3]. Shrivastava et al. propose the usage of replay buffers to stabilize GAN training [19]. This follows from reinforcement learning, where experience replay buffers were famously used by Mnih et al. to stabilize a deep neural network formulation of Q-learning [14]. The replay buffer in our CycleGAN implementation stores previously generated images. The discriminator is then shown these previously generated images during training, encouraging the discriminator to "remember" images the generator network previously produced and be able to discriminate against artifacts it may have "forgotten." Sampling and replacing in the replay buffer is probabilistic – with probability p_{sample}, an example in a batch can be swapped with a sample from the replay buffer.

Two types of noise are added to the discriminator during training as a regularizer. During training, it is possible for the discriminator to dominate the generator. These techniques reduce the efficacy of the discriminator in relation to the generator, making it easier for the generator to compete. Discriminator networks normally classify real images

as true labels α and fake labels β. In the original formulation of the adversarial loss, β is set to 0 while α is set to 1. We instead change these terms to be based on uniform distributions, where $\beta \sim \mathcal{U}(0, 0.3)$ and $\alpha \sim \mathcal{U}(0.7, 1.2)$. This formulation adds noise to the targets of the discriminator networks, effectively weakening the discriminator networks and discouraging them from dominating the generator networks.

Additionally, with random chance p_{flip}, we flip the target α and β such that the discriminator networks are predicting opposite of what it was in the previous batch, i.e. the discriminator classifies real images as fake images and vice versa. This additional noise in the target values for each discriminator further weakens the discriminator networks.

Experiments were also performed with a cyclic consistency loss based on work by Wang and Lin in [20]. Their work observed empirically that the strict cycle consistency constraint in CycleGAN limited generator networks negatively in early epochs. Particularly, CycleGAN's original loss function too heavily emphasizes similarity along the pixel level and not along the feature level. Enforcing cycle consistency along the feature level ensures that features are translated along both domains. In addition to this, the authors weight the cyclic loss by the discriminator output, making the new loss function

$$\begin{aligned}\mathcal{L}_{cyc} = \mathbb{E}_{x\sim p_X}[D_X(x)(\gamma\|f_{D_X}(F(G(x))) - f_{D_X}(x)\|_1 \\ +(1-\gamma)\|F(G(x)) - x\|_1)] \\ +\mathbb{E}_{y\sim p_Y}[D_Y(y)(\gamma\|f_{D_Y}(G(F(y))) - f_{D_Y}(y)\|_1 \\ +(1-\gamma)\|G(F(y)) - y\|_1)]\end{aligned} \tag{5}$$

In this cyclic loss function, γ encodes the trade off between pixel level cyclic consistency and feature level cyclic consistency, and $f_{D_{(\cdot)}}$ is a feature extractor take from the penultimate layer of $D_{(\cdot)}$. This loss function aims to maintain features across image translations, and also downplay the weight of cycle consistency early in training where the generator does not produce realistic images to either domain.

All loss functions in our version of CycleGAN are implemented in a least-squares fashion, similar to [13]. This stabilizes training by allowing both generators and discriminators to minimize a loss function that provides higher gradients when farther from the objective. The loss function for the adversarial loss in our formulation is then

$$\begin{aligned}\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y\sim p_Y}[(D_Y(y) - \alpha)^2] \\ +\mathbb{E}_{x\sim p_X}[(D_Y(G(x)) - \beta)^2]\end{aligned} \tag{6}$$

which can be applied without loss of generality to the other adversarial loss. The objective is now $\max_G \min_{D_Y} \mathcal{L}_{GAN}$. In this formulation, the generator maximizes the difference between the discriminator's output on its images and the target value of β, the label for fake images. The discriminator minimizes the distance between its output on real images w.r.t. α and fake images w.r.t. β. In practice, we modify the generator's loss function to minimize the distance between the opposite label, allowing both the generator and discriminator to perform gradient descent in a stable manner.

The generator architecture used is very similar to the ResNet generator that is used in CycleGAN, itself taking from Johnson et al. in [9]. The generator consists of several reflection pad-convolutional-instance normalization-ReLU blocks that are analogous to an encoder for the image information. This is followed by nine ResNet blocks, formulated exactly as those in [6] and [21]. These blocks act as a transformer, modifying the input in latent space to become an output in the target domain. The decoder differs from CycleGAN in that we do not use transpose convolutions, but rather nearest neighbor upsampling layers. This is done to reduce the presence of checkerboard artifacts in produced images, theorized to be introduced by the striding nature of transpose convolutions [15]. Then the decoder is a series of upsample-reflection pad-convolutional-instance norm-ReLU layers. A tanh layer is applied as the final activation, outputting images normalized between $[-1, 1]$.

The discriminator architecture is the 70×70 PatchGAN architecture used in [8] and [21]. Rather than outputting a singular value per image, this architecture produces a $30 \times 30 \times 1$ output (for input images of size $256 \times 256 \times 3$) that represent the discriminator's output on 70×70 patches of the input image. This discriminator was chosen over the traditional PixelGAN (similar architecture that produces a singular output per image) because PatchGAN allows for spatial encoding of the truth value for different parts of the input image.

5. Experiments

We perform training using the Adam optimizer on both the generators and discriminators, with a learning rate of 0.0002 for both and $\beta_1 = 0.5$, $\beta_2 = 0.999$ for both architectures. Experiments were run with λ_{cyc} equal to 10, a replay buffer of size 50, and p_{flip} equal to 0.05. For experiments with the feature level cycle consistency loss, we use a γ value of 0.05 and linearly anneal it to 0.95, and linearly anneal λ_{cyc} from 10 to 0 over all epochs. We train over 200 epochs with a batch size of 8. We choose a larger batch size than that used in the original CycleGAN implementation to fully utilize our GPU. Otherwise, all hyperparameters are set to their default values from [21].

Training is done with unpaired images from the palm tree synset from ImageNet and automatically generated edge maps from those images. We evaluate training performance both with a test set of palm tree photos and edge maps not used in training, and several tree sketch and tree photo images taken from [18] that were not used in training. Empirical results showed that utilizing these images during train-

ing slowed generator progress towards learning the data distributions, likely because of a large imbalance in the number of examples of the two. Some experiments are also done on the Linden tree synset from ImageNet, though were a secondary point in our investigation and we do not evaluate our model on all the same quantitative metrics with the Linden tree domain.

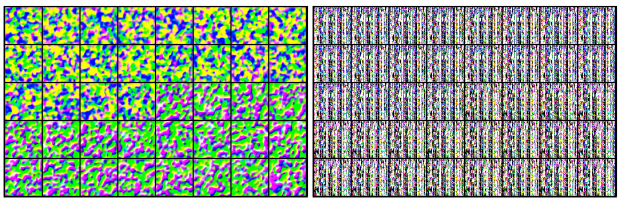

Figure 2. CoGAN generated images of trees after 100 iterations and 99000 iterations.

6. Results

6.1. Evaluation Metrics

The original CycleGAN authors compare their model quantitatively with image segmentation metrics for the CityScapes dataset [4]. FCN scores measure the interpretability of generated images against real images. With the CityScapes dataset, CycleGAN was able to produce photos from instance segmentation labels and vice versa. The produced images can then be compared with per-pixel, per-class, and class Intersection-over-Union metrics to those produced by a standard FCN semantic segmentation network. We are unable to use this metric because our dataset does not contain semantic segmentation labels. This is further complicated by the fact that one of the most commonly used semantic segmentation datasets, Microsoft COCO, does not contain labels for trees [10]. It is difficult to find a pretrained classifier that can create semantic segmentation labels for our generated images to compare against real images.

In addition to FCN scores, the authors use Amazon MechanicalTurk to collect data from human subjects in several trials. This is the ideal measure of photorealism in images, but requires more time than algorithms that output scores and is more subjective in nature. It is often difficult to guard against test subjects who understand what artifacts to look for when examining GAN-produced images, particularly in a computer science department within a university. We instead perform a less thorough qualitative analysis of our images ourselves, noting the strengths and weaknesses of our produced images.

An advantage of utilizing images that come from ImageNet is that there is an abundance of classification models available that were pretrained on ImageNet. This makes it possible to use the Fréchet Inception Distance (FID) as an evaluation metric. The FID compares activation distributions of generated samples vs. real samples on the InceptionV3 network [7]. The FID score is defined as

$$FID = \|\mu_r - \mu_g\|^2 + \mathrm{Tr}\left(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}}\right) \quad (7)$$

where $X_r \sim \mathcal{N}(\mu_r, \Sigma_r)$ and $X_g \sim \mathcal{N}(\mu_g, \Sigma_g)$ are the activations of the pool3 layer in the InceptionV3 network. For this metric, lower is better. FID score is useful for quantifying the quality and diversity of generated images, since it decreases as the number of modes increases and decreases as the presence of similar features increases.

We also examine the quality of the inverse mapping by examining the cycle consistency loss over the whole validation set. A lower cycle consistency loss indicates that little information is lost during translation between domains. To calculate this, we take the mean L1 distance between images and their reconstructions over the whole test set. For this metric, lower is better.

6.2. Baseline

We use CoGAN [11] as a baseline because it is a simpler model that can also learn joint distributions. Our method with CoGAN is to learn the joint distribution of images from p_X and p_Y, where p_X and p_Y are the marginal distributions of tree sketches and tree photos, respectively. CoGAN allows us to learn G_X and G_Y, generative models that allow us to synthesize new images in the distributions X and Y. Since these networks are coupled, i.e. they share parameters in the decoding section of the generator and in the encoding section of the discriminator, we can use both to learn a mapping between X and Y. For a given input x, we must find a noise vector z^* such that $G_X(z^*) \approx x$. The corresponding image y is then just $G_Y(z^*)$.

The baseline method fails to produce realistic images for either domain in our formulation of image-to-image translation. Zhu et al. [21] note that CoGAN was relatively unsuccessful at producing realistic mappings, often falling into modal collapse or producing unrealistic images. Performing the translation technique described above was difficult because we could not achieve convergence with CoGAN on our dataset. We suspect that this is due to the lack of a large dataset, and despite data augmentation techniques that were described earlier, CoGAN failed to learn a meaningful mapping between domains. CoGAN produced remarkably regular outputs, seemingly invariant to the training images we fed into the model. The CoGAN model fails to produce images that can be recognized as realistic trees, with most images qualitatively looking like random noise. Some images of produced outputs at different iterations can be seen in 2.

Translation	CoGAN	CycleGAN	CycleGAN w/ LR	CycleGAN w/ BC	CycleGAN w/ all
edges-to-photos	N/A	**344.02**	363.03	449.05	503.70
photos-to-edges	526.59	306.62	260.57	313.71	**255.29**
sketch-to-photo	538.97	491.06	**434.18**	479.18	526.06
photo reconstruction	N/A	**177.50**	221.17	324.89	324.73
edge reconstruction	N/A	**99.34**	173.22	196.56	196.03

Table 1. FID scores for generated photos of palm trees compared to real photos of palm trees, and generated edge maps compared to real edge maps. Reconstructed images are also compared to their original domains. LR = label regularization, BC = better cycle consistency loss

Reconstruction	CycleGAN	CycleGAN w/ LR	CycleGAN w/ BC	CycleGAN w/ all
edges	**33.23**	36.18	37.21	39.97
photos	101.30	**91.50**	107.37	101.21

Table 2. Mean L1 distance between source images and reconstructed images.

6.3. Analysis

6.3.1 Quantitative Analysis

We compare the original CycleGAN implementation to the CoGAN baseline, in addition to our modified CycleGAN with the various regularization and stabilization techniques described above. In table 1, LR refers to the label regularization technique that involves noisy labels and flipped labels, and BC refers to the addition of feature level cycle discrimination loss introduced by [20]. We compare FID for our CycleGAN variants on a variety of translation tasks, including the generalization task of sketch-to-photo. In addition to FID score, we also examine the cycle consistency loss for the generators, quantifying the quality of the inverse mapping the generators learn. Reconstruction losses are seen in 2. All metrics are measured on the testing sets described previously.

For edge-to-photo translation, we find that the original CycleGAN implementation outperforms all other variants. The addition of label regularization through smoothing and flipping decreased performance significantly The feature-level cycle consistency loss drastically decreases performance. We suspect that since edge maps lack features important to generation of images, such as color, texture, and fine details, a feature-level cycle consistency loss is actually detrimental to performance.

For photo-to-edge translation, our implementation of CycleGAN with label regularization and the feature-level cycle consistency term dominates all other models. There is a significant margin between the variants with label regularization and other models. This can be attributed to regularization's tendency to diversify generated images – the noise added by flipped and noisy labels can reduce overfitting and allow for a more diverse distribution of images.

In the sketch-to-photo generalization task, we see that label regularization again benefits the CycleGAN model. The added noise likely combats overfitting, allowing for more generalization when presented with a sketch rather than an edge map.

We compare the reconstructed images with two metrics: FID score and the cycle consistency loss. The original CycleGAN implementation is better than other models in most situations with both these metrics. Interestingly, the variant with label regularization has a better cycle consistency loss with photos. This indicates that the reconstructed images with vanilla CycleGAN have a higher image diversity than CycleGAN with label regularization, despite being slightly lower quality in reconstruction.

6.3.2 Qualitative Analysis

Our models do not have particularly compelling results for edge-to-photo translation. Qualitatively, even the generated distribution of photos for the best performing CycleGAN variant does not appear to be real. This is evidenced by the clear irregularities present in most images – floating leaves, lack of bark, unrealistic colors, and textures that are clearly repetitive and artificial in nature. There are, however, examples of images that may pass as realistic to an untrained eye 3 on certain datasets. We speculate that the lack of texture in many images and sparse features contributes to this poor performance.

However, the photo-to-edge translation works well qualitatively. The generated sketches match the edge map distribution very well – it appears as if the generator network learned the bilateral filtering and Canny edge detection algorithm. Empirically, the generator producing the edge maps learns the distribution much more quickly that the inverse generator. At times, the textures produced are unrealistic for the parameters of the Canny edge detection algorithm that we select, though look as if a different set of parameters could produce those textures. This good performance is likely because the Canny edge detection algorithm itself is a series of convolutions of kernels over an input image, very similar to the convolutions that the generative network applies to an image.

We also examine generalization of our models from edge-to-photo translation to sketch-to-photo translation. Real sketches fed to the generator produce wildly unrealistic results. At best, the produced image colors the tree sketch with green and colors the sky a shade of sky blue or sunset. A large fraction of images in our palm tree dataset are photos taken at sunset, explaining the bias towards orange and red skies. While sketches from [18] are based on images that are found in ImageNet, they lack the texture and details that an automatic edge detection algorithm provides. In an attempt to better match the conditions of [18], we also perform some experiments with the Linden tree synset, which encompasses many of the tree images in the Sketchy database. The result is still unrealistic, suggesting that the generator depends greatly on the texture and detail of the edge maps to produce realistic images. It's likely that the generator learns to encode information about the photo class within the edges of the generated edge map, and a sketch that does not resemble an edge map will perform poorly in this task. Sketches that possessed more texture details in the leaves and bark were colored more accurately compared to simpler sketches. A common failure mode cited by the original creators of CycleGAN involves the transfiguration of images from one domain to another – CycleGAN often cannot drastically modify the features of the source image to those of the target image. This was often the case for our sketches and edge maps with more sparse edges, where CycleGAN would struggle to fill empty spaces in the source images to create a realistic output in the target domain. Despite this, it is not unreasonable to suggest that a skilled artist could create a very detailed sketch that could be passed to our generator to create a photorealistic image.

6.3.3 Observations during training

During training, our monitoring software suggests that the generator often oscillates between a few modes and fails to maintain a diverse set of outputs. For example, the same source images from different epochs during training often produce target images with different colored skies or colored bark. This can be seen in figure 3. It is difficult to determine the source of these oscillations – the addition of label regularization smooths these oscillations somewhat but results in lower quality images.

We found it difficult to balance the feature level cycle consistency loss with the addition of more regularization through label smoothing and flipping. Since the feature level consistency loss depends on accurate output from the discriminator, it is more desirable to have a competitive discriminator rather than a weakened discriminator. The addition of too much noise in the discriminator's target values weakened the discriminator to the point where the generator was easily able to fool the discriminator with suboptimal data.

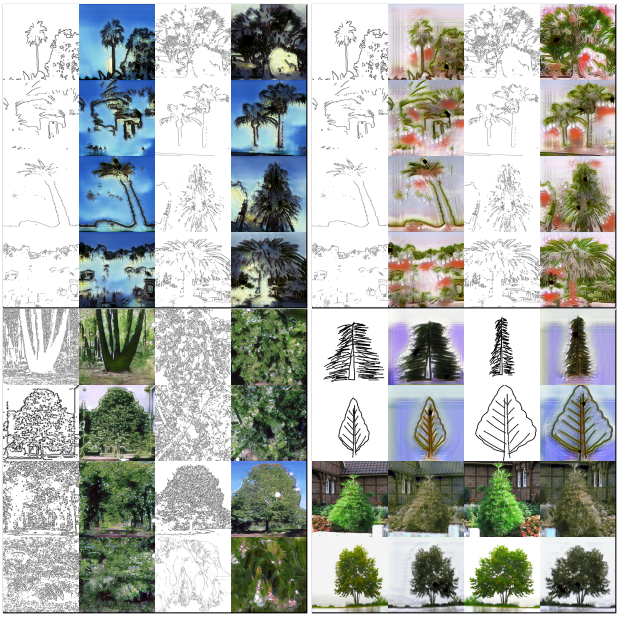

Figure 3. Several images produced by the generators. The top left and right images demonstrate the oscillation of modes for the sky color after just 1 epoch. The bottom left image shows the ability of CycleGAN to produce photorealistic images with the Linden tree dataset. The bottom right image shows qualitative performance of the generator on actual sketches, and also an example of the inverse mapping from photo to edge to photo.

Experiments with different types of architectures also provided some interesting results. The transition from transpose convolution to resize-convolution noticeably reduced the presence of artifacts that resulted from the stride of the convolution. Preliminary runs were done with PixelGAN over PatchGAN, though the loss of spatial information in the discriminator output that resulted from this caused a massive drop in image quality. The change to PatchGAN greatly improved our image quality, and this sufficiently powerful discriminator was able to balance the very expressive ResNet generator network well.

7. Conclusion

In conclusion, we demonstrate generative networks' ability to produce semi-photorealistic images of trees from automatically generated edge maps. The generative models do not generalize very well to the sketches we provide, but with a sufficiently detailed sketch it is possible to generate photorealistic images. The regularization techniques we employ are mildly effective in improving model performance on this task, though when combined are less effective, likely because the two techniques are themselves contrarian in nature. Our best performing model for the

generalization task was CycleGAN with label regularization through smoothing and flipping.

Future work in this domain should explore the translation of approximations of images to photos. Work in this niche, like work done by Park et al. in [16], can vastly improve rapid prototyping in media and other creative fields. Use of this technology for semantic segmentation also presents interesting use cases in robotics and autonomy in applications like self-driving cars and other autonomous vehicles.

References

[1] A. Bansal, S. Ma, D. Ramanan, and Y. Sheikh. Recycle-gan: Unsupervised video retargeting. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[2] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, Nov 1986.

[3] S. Chintala, E. Denton, M. Arjovsky, and M. Mathieu. How to train a gan? tips and tricks to make gans work. `https://github.com/soumith/ganhacks`, 2016.

[4] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685, 2016.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[7] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, abs/1706.08500, 2017.

[8] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016.

[9] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, 2016.

[10] 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.

[11] M. Liu and O. Tuzel. Coupled generative adversarial networks. *CoRR*, abs/1606.07536, 2016.

[12] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 700–708. Curran Associates, Inc., 2017.

[13] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, and Z. Wang. Multi-class generative adversarial networks with the L2 loss function. *CoRR*, abs/1611.04076, 2016.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[15] A. Odena, V. Dumoulin, and C. Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[16] T. Park, M. Liu, T. Wang, and J. Zhu. Semantic image synthesis with spatially-adaptive normalization. *CoRR*, abs/1903.07291, 2019.

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.

[18] P. Sangkloy, N. Burnell, C. Ham, and J. Hays. The sketchy database: learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics (TOG)*, 35(4):119, 2016.

[19] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. *CoRR*, abs/1612.07828, 2016.

[20] T. Wang and Y. Lin. Cyclegan with better cycles.

[21] 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.

[22] J. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman. Toward multimodal image-to-image translation. *CoRR*, abs/1711.11586, 2017.

8. Contributions and Acknowledgements

Kevin Wang performed the majority of the background research, and found the label regularization techniques that we applied to our implementation. He also contributed to selecting the dataset. He also ensured that the methods section was correct mathematically. Cedrick Argueta wrote a majority of the CycleGAN implementation and preprocessing code for the edge detector. He also wrote code for interacting with the different datasets. He also found evaluation metrics suitable to the task used them to generate quantitative results. Both authors contributed equally to training the models on Google Cloud and to interpreting the results of the project.

9. Code

The public repository for the project can be found here. The code for the replay buffers is adapted from the official CycleGAN implementation here. Many existing open source implementations of CycleGAN were difficult to read/did not implement the same loss functions we attempt to minimize, so we created the training and testing code ourselves. We also take from the official implementation the code for learning rate scheduling, along with cues for what our loss functions should look like in the least squares formulation (i.e. instead of implementing a least squares difference ourselves, we learned that it was possible to use cri-

terion in PyTorch). The code for FID score comes from this repo, where it is directly used to calculate our FID scores. The code for the CoGAN baseline is from this open source repo, only slightly adapted to work with our datasets.