On Applications of GANs and Their Latent Representations

Cameron Fabbri
Computer Science and Engineering
University of Minnesota
Minneapolis, MN 55455
fabbr013@umn.edu

Abstract

This report describes various applications of Generative Adversarial Networks (GANs) for image generation, image-to-image translation, and vehicle control. With this, we also investigate the role played by the computed latent space, and show various ways of exploiting this space for controlled image generation and exploration. We show one pure generative method which we call AstroGAN that is able to generate realistic images of galaxies from a set of galaxy morphologies. Two image-to-image translation methods are also displayed: StereoGAN, which is able to generate a pair of stereo images given a single image; Underwater GAN, which is able to restore distorted imagery exhibited in underwater environments. Lastly, we show a generative model for generating actions in a simulated self-driving car environment.

1 Introduction

Vision is a very powerful tool used across many different academic and industrial communities including healthcare, natural sciences, entertainment, and commerce. Images are able to capture various levels of abstraction that are easy for a human to decipher, but may be more challenging for a machine. For example, given an image of some scene, a human may be able to classify the scene as "dangerous" or "not dangerous", despite the many abstract details in the image causing them to make that decesion. Classically, for a machine to perform the same task, it must be able to extract explicit details from the image that would cause it decide on one or the other.

With the ever growing amount of data, it is important to enable automated approaches for analyzing and extracting useful information from these images. Recent advances in deep learning [14] has enabled a variety of successful approaches towards doing just this. Machines are now able to, with close or better than human accuracy in some areas, reason and extract meaningful details and abstract concepts from images given a sufficiently large amount of data, such as image classification [13].

While this may appear to be a purely discriminative task, we present three reasons for which a *generative* model can be of use. Some generative processes are mostly methods to be used early in the automation pipeline, with the goal of enhancing the discriminative methods being used to extract information. For each case we point to a specific application able to overcome and address the issue at hand.

Missing Data Although we live in the age of (seemingly endless) information, there are still environments and situations in which certain data is underrepresented. Thinking about image datasets in a probabilistic sense as samples coming from some unknown distribution, none if any are uniform. In other words, many times we have outliers in datasets, and these outliers in fact may be the most interesting data points. There are many environments, especially in the natural sciences, in which it is

very difficult to obtain a sufficient amount of data across all types of subdomains. These difficulties may arise from simply the lack of data naturally available, or the high cost that comes with collecting it.

The field of astronomy is very interested in capturing images of galaxies, as these images capture detailed morphologies able to give clues about the evolution such galaxies [16, 17]. Furthermore, due to the finite speed of light, images of galxies at different distances are able to capture a younger universe. While many of these morphological features are due to the properties exhibited by the galaxy (*e.g., the presence of spiral arms*), others may be due to the line of sight. This line of sight also plays a major role in missing data, as because we cannot simply travel to another location of the universe to capture a different view, we are stuck with a single view for each galaxy. If we are to capture a galaxy with extremely rare morphological properties, then we unfortunately have to deal with the fact that this may be the only viewpoint available. Due to many machine learning algorithms requiring a vast amount of data to learn from, this makes it difficult to train these algorithms to automatically extract information from these sorts of outliers. In Section 6 we display a method to generate images of galaxies given a set of morphological features.

A very different form of missing data may come from sensor failure or sensor deprivation. In the field of robotics, stereo vision is a commonly used form of perception. Given a pair of stereo images, as well as the camera intrinsics and extrinsics, one is able to effectively calculate the distance to an object of interest in the frame. However, given a robot with only a single camera, or a case in which one of these cameras fail, this calculation is unable to be performed. Section 4 shows a method able to create a pair of stereo images given only a single image. Given the image captured by the left camera, we are able to generate what would be seen by the right camera, and vice versa. This application is general, and can be applied to systems with only a single camera, allowing such systems to hallucinate a second camera.

Corrupted Data Many environments are naturally noisy in the visual domain. This noise may come from the environment itself, or manmade alterations aimed at improving vision (such as night vision or infrared). In either case, this noise causes many vision related tasks such as classification, tracking, or segmentation to suffer. For this reason, we are interested in using generative modeling to enhance these images such that they can be used further down the autonomy pipeline. Section 5 focuses on the underwater domain, for which distortion is especially prevalent.

In the underwater domain, light refraction, absorption, and scattering from suspended particles can greatly affect optics. Due to the absorption of red wave lengths by the water, images tend to have a blue or green hue. This effect worsens as one goes deeper and more red light is absorbed. This distortion is extremely non-linear in nature, and is affected by a large number of factors, such as the amount of light present, the amount of particles in the water, the time of day, and the camera being used. Not only may this distortion may cause difficulty in vision-based tasks, visual inspection by humans becomes more difficult as the quality of the images degrades.

Correct Answers Tasks such as classification or regression naturally have one correct answer or label. However, in some applications, there may be more than one viable choice. Consider playing a video game in which you are driving a car around in a city. The field of Reinforcement Learning is interested in finding an optimal policy in which to do so, in order to maximize some reward. However, we argue that depending on the task, there may not be an optimal policy, and therefore no optimal action to take at a certain timestep.

We look into the task of automatically driving a car in a simulation with absolutely no goal, other than to mimic a human player. This leaves us with a situation in which for a single input (*e.g.,* an intersection) in which more than one choice may be correct (*e.g.,* turning left or right). Rather than constrain the solution space by using regression, we formulate the problem using GANs in order to generate a realistic action given a stack of n frames as input.

Section 2 gives a brief overview of the core design behind each method. Section 3 gives an overview of our network architecture used in StereoGAN and UGAN. Section 4 presents StereoGAN, a method to generate a stereo pair from a single image. Section 5 presents Underwater GAN, a method to enhance and restore distorted underwater images. Section 6 presents AstroGAN, which is able to generate galaxies conditioned on a set of morphological features. Section 7 presents Driving GAN (DGAN), which aims to produce actions conditioned on frames. Finally, we conclude in Section 8

2 Background

Here we give a very brief background on Generative Adversarial Networks (GANs) and some of their variants. We do not make any theoretical claims or improvements; rather, we explore applications in which GANs proove to be very useful, and in some cases necessary over previous generative models. Readers are directed towards [8, 1, 20, 2, 22, 9] for further details in the theoretical domain.

Generative Adversarial Networks Generative Adversarial Networks (GANs) [8] represent a class of generative models that are based on game theory. Given two differentiable functions modeled as neural networks, a generator G attempts to generate data points given input noise that are able to fool a discriminator D. The discriminator is given either real data points or data points produced by G, and aims to classify them as real or fake. This is formally defined as the following minimax function:

$$\min_G \max_D \mathbb{E}_{x\sim\mathbb{P}_r}[log\ D(x)] + \mathbb{E}_{z\sim\mathbb{P}_z}[log\ (1 - D(G(z)))] \tag{1}$$

where $\mathbb{P}_z$ is a prior on the input noise (*e.g.*, $z \sim \mathcal{N}(0, 1)$) and $\mathbb{P}_r$ is the true dataset.

Conditional GANs Conditional GANs (cGANs) [18] introduce a simple method to give varying amounts of control to the image generation process by some extra information y (e.g a class label). This is done by simply feeding y to the generator and discriminator networks, along with z and x, respectively. The new objective function then becomes:

$$\min_G \max_D \mathbb{E}_{x,y\sim\mathbb{P}}[log\ D(x, y)] + \mathbb{E}_{z\sim\mathbb{P},y\sim\mathbb{P}}[log\ (1 - D(G(z, y)))] \tag{2}$$

Wasserstein GAN It has been shown that even in very simple scenarios the original GAN formulation is very unstable. The discriminator as a classifier may not supply useful gradients for the generator [2], and the vanishing gradient problem stops G from learning. On the other hand, the EM distance does not suffer from these problems of vanishing gradients. The EM distance or *Wasserstein-1* is defined as:

$$W(\mathbb{P}_r, \mathbb{P}_g) = inf_{\gamma\in\prod(\mathbb{P}_r,\mathbb{P}_g)}\mathbb{E}_{(x,y)\sim\gamma}[||x - y||] \tag{3}$$

where $\prod(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of joint distributions $\gamma(x, y)$ whose marginals are $\mathbb{P}_r$ and $\mathbb{P}_g$, respectively [2]. Because the infimum is very troublesome, [2] instead proposes to approximate W given a set of K-Lipschitz functions f by solving the following:

$$\max_{w\in W} \mathbb{E}_{x\sim\mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z\sim\mathbb{P}_z}[f_w(G_\theta(z))] \tag{4}$$

Improved Wasserstein GANs In order to enforce the Lipschitz constraint without clipping the weights of the discriminator, [9] instead penalizes the gradient, leading to a WGAN with gradient penalty (WGAN-GP). More formally, a function is 1-Lipschitz only if it has gradients with a norm of 1 almost everywhere. In order to enforce this, WGAN-GP directly constrains the norm of the discriminator's output with respect to its input. This leads to the new objective function:

$$\max_{w\in W} \mathbb{E}_{z\sim p(z)}[f_w(G_\theta(z))] - \mathbb{E}_{x\sim\mathbb{P}_r}[f_w(x)] + \lambda\mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}[(||\nabla_{\hat{x}} f_w(\hat{x})||_2 - 1)^2] \tag{5}$$

where $\mathbb{P}_{\hat{x}}$ is defined as sampling uniformaly along straight lines between pairs of sampled from the true data distribution, $\mathbb{P}_r$, and the distribution assumed by the generator, $\mathbb{P}_g = G_\theta(z)$. This is motivated by the intracability of of enforcing the unit gradient norm constraint everywhere. Because the optimal discriminator consists of straight lines connecting the two distributions (see [9] for more details), the constraint is enforced uniformaly along these lines. As with the original WGAN, the discriminator is updated n times for every 1 update of the generator.

Before turning to our applications, we go over the architecture of the generator used in StereoGAN (Section 4) and UGAN (Section 5). The discriminator used in all three applications is the same, and we utilize the Improved Wasserstein GAN formulation.

3 Network Details

Generator At their core, StereoGAN and UGAN are image-to-image translation methods. As such, their main differences lie within the task at hand, leaving the design of the architecture to remain the same. Our generator network is a fully convolutional encoder-decoder, similar to the work of [11], which is designed as a "U-Net" [21] due to the structural similarity between input and output. Encoder-decoder networks downsample (encode) the input via convolutions to a lower dimensional embedding, which is then upsampled (decode) via transpose convolutions to reconstruct an image. The advantage of using a "U-Net" comes from explicitly preserving spatial dependencies produced by the encoder, as opposed to relying on the embedding to contain all of the information. This is done by the addition of "skip connections", which concatenate the activations produced from a convolution layer i in the encoder to the input of a transpose convolution layer $n - i + 1$ in the decoder, where n is the total number of layers in the network. Each convolutional layer in our generator uses kernel size 4×4 with stride 2. Convolutions in the encoder portion of the network are followed by batch normalization [10] and a leaky ReLU activation with slope 0.2, while transpose convolutions in the decoder are followed by a ReLU activation [19] (no batch norm in the decoder). Exempt from this is the last layer of the decoder, which uses a TanH nonlinearity to match the input distribution of $[-1, 1]$. Recent work has proposed Instance Normalization [27] to improve quality in image-to-image translation tasks, however we observed no added benefit.

Discriminator Our fully convolutional discriminator is modeled after that of [20], except no batch normalization is used. This is due to the fact that WGAN-GP penalizes the norm of the discriminator's gradient with respect to each input individually, which batch normalization would invalidate. Our discriminator is modeled as a PatchGAN [11, 15], which discriminates at the level of image patches. As opposed to a regular discriminator, which outputs a scalar value corresponding to real or fake, our PatchGAN discriminator outputs a $32 \times 32 \times 1$ feature matrix, which provides a metric for high-level frequencies.

4 StereoGAN

The use of stereo vision is very common in many computer vision applications, especially in the field of robotics. By utilizing knowledge about the camera parameters, one is able to extract 3D information for a given scene from a pair of images. Stereo matching, also known as disparity mapping, aims to produce a depth map for an image to extract the depth for each pixel. Classically this is done using some feature matching algorithm such as Sift or ORB.

Recently there have been machine learning approaches to the problem of stereo matching [31, 12]. Often dealing with hardware may be an issue, whether it be from cost, maintenance, or design. Towards this end, many techniques have been geared towards depth map prediction from a single image [5, 6]. Most similar to our approach is the work of [7], who proposed a depth estimation method by generating a disparity image. Given a pair of ground truth stereo images they aim to generate the right image given the left and introduce a novel loss in order to enforce consistency between disparities. The work of [29] has a similar approach, but for the task of creating stereo pairs for 3D movies.

These methods tailored their loss functions towards the task of depth estimation. We take a different approach. Our goal is to be able to generate a stereo pair given a single image for use in any task involving stereo vision. Different from past methods, we train a deep convolutional neural network to generate the left image given the right *as well as* generate the right image given the left. As discussed in our experiments, being able to generate both shows to capture interesting 3D properties within the model that can be exploited.

Approach As the name suggests, we use a Generative Adversarial Network as our generative model. GANs are able to produce sharp images not exhibited when using a pixel-wise metric such as L_2. Past methods such as [7] that used a loss tailored to disparity were not interested in the visual output, so as long as the generated image improved the disparity calculation, it was good enough. We are interested in this as a general approach for computing stereo images, so we do care about the image quality.

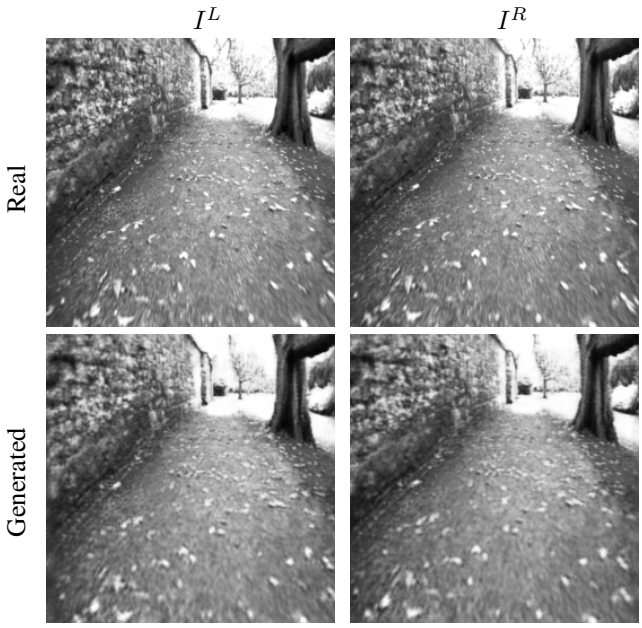

Figure 1: Samples generated by **StereoGAN** compared with ground truth. The first column displays I^L, and the second column displays I^R. Top: group truth. Bottom: generated.

Generator Our architecture is similar to siamese networks [4], except we do not use shared weights in the generators. Each generator is modeled from the pix2pix architecture [11]. We use two generators: G_1, which takes as input a right image and outputs a left, and G_2, which takes as input a left image and outputs a right. Let I^L be an image seen from the left camera, and I^R be an image seen from the right camera. Our two generators are then defined as:

$$I^{L'} = G_1(I^R; \theta_{G_1}) \tag{6}$$

$$I^{R'} = G_2(I^L; \theta_{G_2}), \tag{7}$$

where $I^{L'}$ and $I^{R'}$ are the generated left and right images respectively, and θ_{G_1}, θ_{G_2} represent the weight parameters for each generator.

Discriminator There are multiple ways we could design our discriminator. Unlike a nonconditional GAN in which the discriminator simply takes as input a single image, we want to condition our generated image on a second image, namely either the left or right in the stereo pair (a true image). We decided on simply stacking the channels, left image on top of the right. We need to provide D with real samples as well as fake samples coming from G. Because we have two generators, we need to send D fake samples from both. This is shown formally in Equation 8.

We use the Improved Wasserstein method as discussed in Section 2. For sake of space and simplicity in notation, we define the gradient penalty as $\mathcal{L}_{GP} = \lambda_{GP}\mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}[(||\nabla_{\hat{x}}D(\hat{x})||_2 - 1)^2]$, where $\mathbb{P}_{\hat{x}}$ is defined as sampling along straight lines between points from the data distribution $\mathbb{P}_r$ and the generator distribution $\mathbb{P}_g$. We also consider the L_1 loss as well, defined as $\mathcal{L}_{L_1} = \frac{1}{2}(||I^L - I^{L'}||_1 + ||I^R - I^{R'}||_1)$. Our final objective function is then:

$$\mathcal{L}_{SGAN}(G, D) = \mathbb{E}[D(I^L \oplus I^R)] - \frac{1}{2}\mathbb{E}[D(I^L \oplus I^{R'}) + D(I^{L'} \oplus I^R)] + \mathcal{L}_{GP} + \mathcal{L}_{L_1} \tag{8}$$

where $\oplus$ is concatenation along the image channels, and $I^{L'}$, $I^{R'}$ are defined in 6 and 7 respectively. We use a $\frac{1}{2}$ in Equation 8 because we are receiving two losses from D on our generated data, so we want to average the loss between the two.

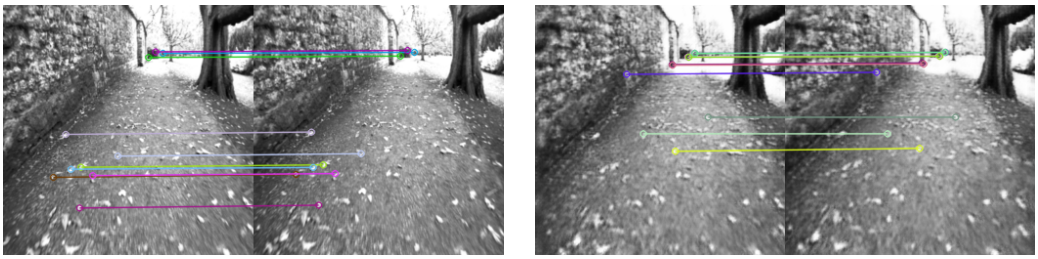

Figure 2: Feature matching using an ORB detector. Top Row: Left and right real images. Bottom Row: Left and right generated images.

Experiments We display the results of the generated images on a held out test set. We use the New College Dataset [25], which consists of stereo images taken at New College, Oxford. We show a qualitative comparison of our generated images with the ground truth, as well as a comparison of running a stereo match algorithm. Figure 1 shows a comparison of our generated images with ground truth. Where as a simple homography would be able to rotate the image, it is not able to capture translation. Our method however, is able to capture translation, as well as inpainting along the edges of the images.

Because our method is able to generate both left and right images, an interesting experiment we can run is the case of *already having a stereo pair*, and generating images outside of the camera's field of view. Consider Figure 3. The green lines can be though of as the true stereo images, coming from for example a mobile robot surveying a scene. If we are interested in something like structure from motion, then having more than just two images is desirable. Instead of using G_1 to generate I^L from I^R, we can instead use I^L as an input, and compute a new I^L (and vice versa with G_2). This is shown explicitly in Figure 4.

A more comprehensive metric is to compare to past methods for computing disparity, but we leave that for our future work.

Figure 3: Diagram of generating images outside of the camera's field of view. Green lines represent the true viewpoints of the two cameras, and the blue lines are the synthesized images. $I^{L_2^{'}} = G_1(I^{L_1})$ and $I^{R_2^{'}} = G_2(I^{R_1})$

Figure 4: Images generated outside of the camera's field of view. The images correspond to those in Figure 3. The two center images are the true stereo pair, and the two images on the ends are generated. Our method is able to inpaint outside of the scene. For example, the wall in front of the cart in the far left image was generated even though it cannot be seen in the two original images. This transformation can be seen easier in video format: `https://i.imgur.com/QOS4au4.gif`

5 Underwater GAN

Underwater images distorted by lighting or other circumstances lack ground truth, which is a necessity for previous colorization approaches [30]. Furthermore, the distortion present in an underwater image is highly nonlinear; simple methods such as adding a hue to an image do not capture all of the dependencies. Physics based models have been designed in order to capture this distortion, but are unable to generalize given the extreme differences two separate environments may exhibit [23]. Towards this end of capturing the extreme nonlinear distortions without designing environment specific models, we use CycleGAN [32] as a distortion model in order to generate paired images for training. CycleGAN is an unsupervised image-to-image style translation method (*i.e.,* it does not need paired samples). Given two domains X and Y, CycleGAN learns a mapping $G : X \rightarrow Y$ such that images sampled from $G(X)$ appear to have come from Y, as well as a mapping $F : Y \rightarrow X$. In order to ensure the translated image still contains properties from the original image, a constraint on the model is the *cycle consistency loss*, $F(G(X)) = X$ (and vice versa). Given a domain of underwater images with no distortion, and a domain of underwater images with distortion, CycleGAN is able to perform style transfer. Given an undistorted image, CycleGAN distorts it such that it appears to have come from the domain of distorted images. These pairs are then used in our algorithm for image reconstruction and enhancement.

Dataset Generation Depth, lighting conditions, camera model, and physical location in the underwater environment are all factors that affect the amount of distortion an image will be subjected to. Under certain conditions, it is possible that an underwater image may have very little distortion, or none at all. We let I^C be an underwater image with no distortion, and I^D be the same image with distortion. Our goal is to learn the function $f : I^D \rightarrow I^C$. Because of the difficulty of collecting underwater data, more often than not only I^D or I^C exist, but not both.

To circumvent the problem of insufficient image pairs, we use CycleGAN to generate I^D from I^C, which gives us a paired dataset of images. Given two datasets X and Y, where $I^C \in X$ and $I^D \in Y$, CycleGAN learns a mapping $F : X \rightarrow Y$. Figure 5 shows paired samples generated from CycleGAN. From this paired dataset, we train a generator G to learn the function $f : I^D \rightarrow I^C$. It should be noted that during the training process of CycleGAN, it simultaneously learns a mapping $G : Y \rightarrow X$, which is similar to f. In Section 5, we compare images generated by CycleGAN with images generated through our approach. Because paired image-to-image translation is a simpler problem, our method is able to outperform CycleGAN for this task.

Figure 5: Paired samples of ground truth and distorted images generated by CycleGAN. Top row: Ground truth. Bottom row: Generated samples.

Methodology We use the Improved Training of WGAN as described in Section 2. Our network architecture is described in Section 3. Conditioned on a distorted image I^D, the generator is trained to produce an image to try and fool the discriminator, which is trained to distinguish between the true non-distorted underwater images and the supposed non-distorted images produced by the generator. Given $I^C \in X$, $I^D \in Y$, and G and D both deep neural networks, our objective then becomes

$$\mathcal{L}_{WGAN}(G, D) = \mathbb{E}[D(I^C)] - \mathbb{E}[D(G(I^D))] + \lambda_{GP}\mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}[(||\nabla_{\hat{x}}D(\hat{x})||_2 - 1)^2], \tag{9}$$

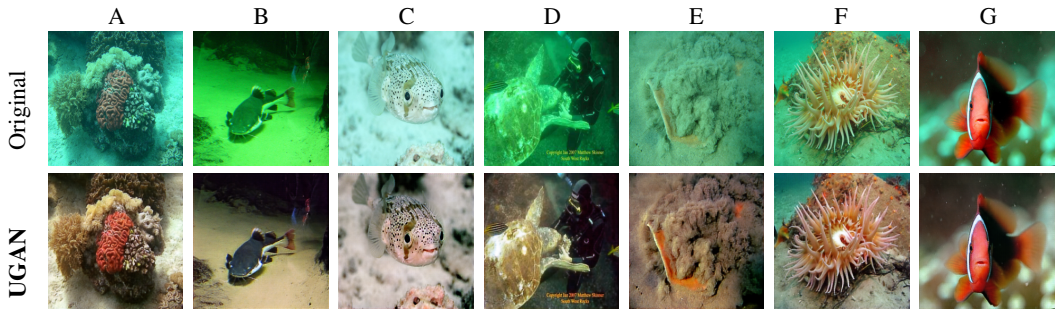

Figure 6: Samples from our ImageNet testing set. The network can both recover color and also correct color if a small amount is present.

where $\mathbb{P}_{\hat{x}}$ is defined as samples along straight lines between pairs of points coming from the true data distribution and the generator distribution, and λ_{GP} is a weighing factor. In order to give G some sense of ground truth, as well as capture low level frequencies in the image, we also consider the $L1$ loss

$$\mathcal{L}_{L1} = \mathbb{E}[||I^C - G(I^D)||_1]. \tag{10}$$

Combining these, we get our final objective function for our network, which we call **Underwater GAN (UGAN)**,

$$\mathcal{L}^*_{UGAN} = \min_G \max_D \mathcal{L}_{WGAN}(G, D) + \lambda_1 \mathcal{L}_{L1}(G). \tag{11}$$

Experiments We experimented on distorted underwater images taken from a test set. Given that these images do not have ground truth, quantitative results may be difficult to acquire. Figure 6 display qualitative results. Not only is UGAN able to restore completely distorted images, it is also able to preserve correct color and restore only parts of the image that have been distorted (Column G).

For a quantitative result, we look toward local image patch statistics, specifically the mean and standard deviation of a patch. The standard deviation gives us a sense of blurriness because it defines how far the data deviates from the mean. In the case of images, this would suggest a blurring effect due to the data being more clustered toward one pixel value. Table 1 shows the mean and standard deviations of the RGB values for the local image patches seen in Figure 8. Despite qualitative evaluation showing our methods are much sharper, quantitatively they show only slight improvement over CycleGAN. Other metrics such as entropy are left as future work.

Latent Exploration Here, we discuss our insights to the inner workings of the model. With a normal autoencoder, the latent embedding would contain all of the information about the image. One can perform certain operations on this embedding, and see a change in the pixel space, as seen in Section 6. However, UGAN makes use of skip connections due to the spatial similarity between input and output. What this means is the latent embedding is no longer forced to contain every bit of information about the input.

Intuitively, one would expect the skip connections to contain information dealing with the image structure, and the embedding to contain color content. However, we concluded this is *not* the case.

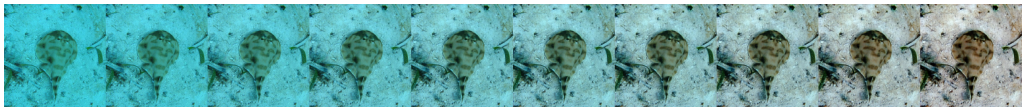

Figure 7: Interpolation in the image space. The far left is the original image, and the far right is corrected by UGAN. These intermediate images may be used as more training samples.

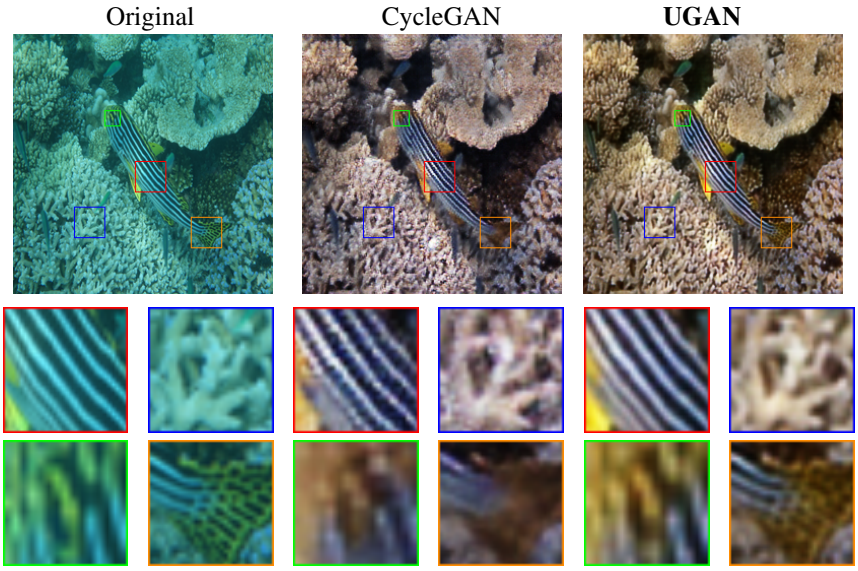

Figure 8: Local image patches extracted for qualitative and quantitative comparisons, shown in Table 1. Each patch was resized to 64×64, but shown enlarged for viewing ability (best seen in PDF).

Our experiment was set up as follows. We took a clean underwater image I^C, and distorted it using CycleGAN to create I^D. Then we used these as input to the generator network, and saved out their latent embeddings. Formally, the embedding for I^C and I^D respectively is $\mathbf{e}_C \in \mathbb{R}^{512}$ and $\mathbf{e}_D \in \mathbb{R}^{512}$. Linear interpolation was used to interpolate between $\mathbf{e}_C$ and $\mathbf{e}_D$. These values were sent through the decoder, along with the respective skip connections. However, there was *no change* in the output image. We then sampled randomly from a Gaussian distribution in a range $[-1, 1]$ to use as the embeddings, and found the same conclusion. This shows that the embedding plays a minimal role, and most information is contained elsewhere in the model. What this insight provides us is a way to try and reduce the number of model parameters without losing accuracy.

Table 1: Mean and Standard Deviation Metrics

Method/ Patch	Original	CycleGAN	**UGAN**
Red	0.43 ± 0.23	0.42 ± 0.22	0.44 ± 0.23
Blue	0.51 ± 0.18	0.57 ± 0.17	0.57 ± 0.17
Green	0.36 ± 0.17	0.36 ± 0.14	0.37 ± 0.17
Orange	0.3 ± 0.15	0.25 ± 0.12	0.26 ± 0.13

Interestingly, we can linearly interpolate in the image space. Figure 7 shows interpolation between two images. While not particularly useful, it is still visually interesting. An idea for future work is to use these interpolants as training points in order to capture a wider variety of distortions.

6 AstroGAN

This section presents **AstroGAN**, a generative model able to generate new, unseen images of galaxies given a morphological description. This method allows astronomers to generate a galaxy with a known set of morphological features in *different viewpoints*. Unlike many other natural images, the field of astronomy contains images of galaxies in which only a single view is available due to the extreme distance between us and the observed galaxy. Two different galaxies may contain the same morphology, but appear to be visually different depending on the line of sight. These differences can cause learning algorithms to suffer, as many state of the art machine learning techniques now require vast amounts of data. **AstroGAN** provides a step towards artifically generating galaxies with a specific morphology in order to improve or verify machine classifiers. A natural extension to our model is the introduction of an encoder network, in order to allow one to perform modifications to a specific galaxy of interest. We display our methods on two different datasets.

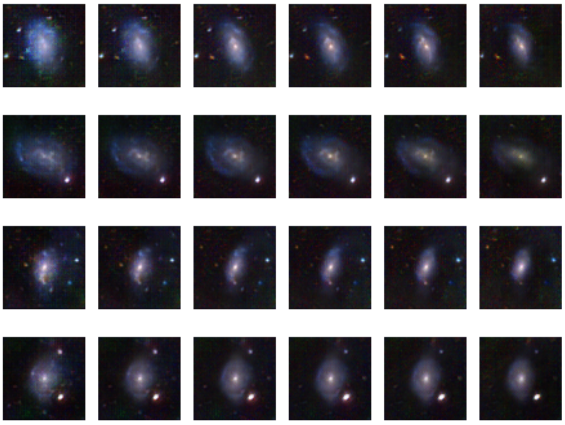

Figure 9: Interpolation along the redshift attribute using the EFIGI dataset. Each row contains the same z and (first four) y values, and linearly interpolates along the redshift attribute. Despite the model never seeing a redshift value outside of $[2.635e\text{-}5, 0.08245]$, it is able to smoothly interpolate between $[0.005, 0.1]$.

EFIGI The first dataset we use to evaluate our model is the EFIGI ("Extraction de Formes Idealisées de Galaxies en Imagerie") catalogue [3], which contains 4458 galaxies with detailed morphologies classified by 10 expert astronomers. Each morphology contains 16 shape attributes, plus one attribute accounting for redshift. From these, including the redshift attribute, we choose those with the most visual impact, leaving us with 5 attributes. Further details on the attribute values can be found in [3]. We use 4058 of these in our training set, and hold out 400 for our test set.

Arm Strength The arm strength attribute measures the relative strength of the spiral arms in terms of the flux fraction relative to the entire galaxy. This value ranges from 0 to 1 in increments of 0.25, with 0 having very weak or no spiral arms, and 1 having the highest contribution of spiral arms.

Arm Curvature The arm curvature attribute measures the average intrinsic curvature of the spiral arms. This value ranges from 0 to 1 in increments of 0.25, with 0 having wide open spiral arms, and 1 having the tightly wound spiral arms.

Visible Dust The visible dust attribute measures the strength of the features revealing the presence of dust, including obscuration and diffusion of star light. This value ranges from 0 to 1 in increments of 0.25, with 0 having no dust, and 1 having a high amount of dust.

Multiplicity The multiplicity attribute quantifies the abundance of galaxies in the neighborhood of the central galaxy. This value ranges from 0 to 1 in increments of 0.25, with 0 having no nearby galaxies, and 1 having four or more nearby galaxies.

Redshift The redshift attribute measures the cosmological redshift exhibited by the galaxy. This is a *continuous* value with a range of $[2.635e\text{-}5, 0.08245]$.

While the first four of these attributes are able to be discretized, redshift is inherently continuous. The combination of discrete and continuous attributes displays the flexibility our model exhibits in capturing these properties. An important factor in the detection of morphological features is the effect redshift can have on a galaxy. Human lifespans prevent viewing the same galaxy at vastly different redshift values, therefore determining what certain morphological features may be visible as a galaxy evolves is not straightforward. Figure 9 displays linear interpolation along the redshift attribute, while the rest of the galaxy structure is kept the same. Visually, one can see the change in morphological features, as the galaxy begins to dim at higher redshift values. Although the model is never given the same galaxy at different redshift values (because of the impossibility of acquiring that data), there is enough information contained in the individual data points for the model to learn its effect on the visibility of morphological features. Surprisingly, the model was able to extrapolate the effect as the attribute value varied outside the range actually present in the training data (e.g., Fig. 9).

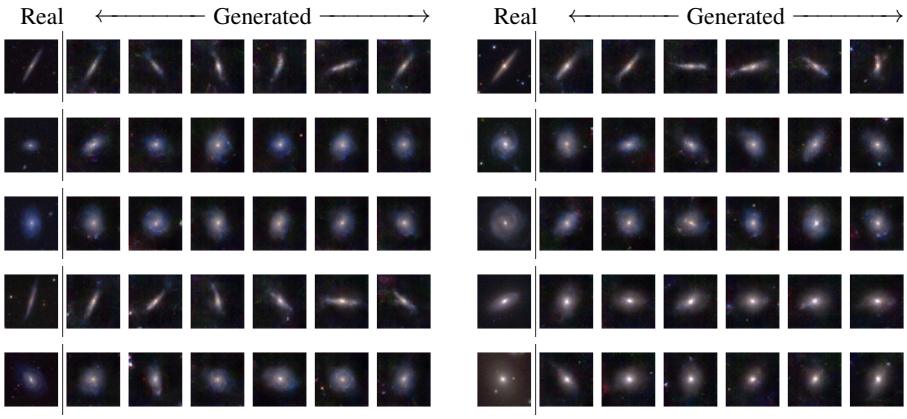

Figure 10: Two sets of samples from the EFIGI dataset and generated samples using our GAN. Samples in the left columns are real images from the true dataset. For the generated samples, each row uses the same y value (attribute vector) as the true image in the left column, and each column uses a different z value. The attributes y used to generate were not available during training, and held out in a separate test set.

Figure 10 displays novel galaxies generated using attributes from the test set. We compare these generated galaxies with true images coming from the test set with the *same attribute*. Qualitative results show that the generated galaxies all exhibit similar morphological features, while also displaying a fair amount of diversity (*i.e.* the generated galaxies are not the same exact image). Additionally, we show interpolation along individual attributes contained in y in Figure 12. The smooth interpolation along the manifold allows for new data not observed in the train or test set to be generated.

Galaxy Zoo The Galaxy Zoo project [17] is an ensamble of citizen science projects that aim to use the crowd to assist in the morphological classification of galaxies. In a typical project, each galaxy is inspected by multiple volunteers, who answer a number of questions pertaining the object's appearance. The questions are organized in a tree-based structure, with the first questions dividing the galaxies into broad morphological classes before subsequent questions look at increasingly detailed aspects of a galaxy's appearance. An example of a Galaxy Zoo decision tree is shown in visual form in Figure 1 of Simmons et al. (2017). As a result of the full set of questions, a continuous attribute $y \in \mathbb{R}^{37}$ is associated with each galaxy, and the degree of consistency between the classifications provided by volunteers provides a measure of the precision of their aggregate classification (see [28] for more information). Of the $61,578$ images we use $60,000$ for training and $1,578$ for testing. Figure 11 displays instances generated from attributes contained in our test set. The network architecture is identical of that used on the EFIGI dataset.

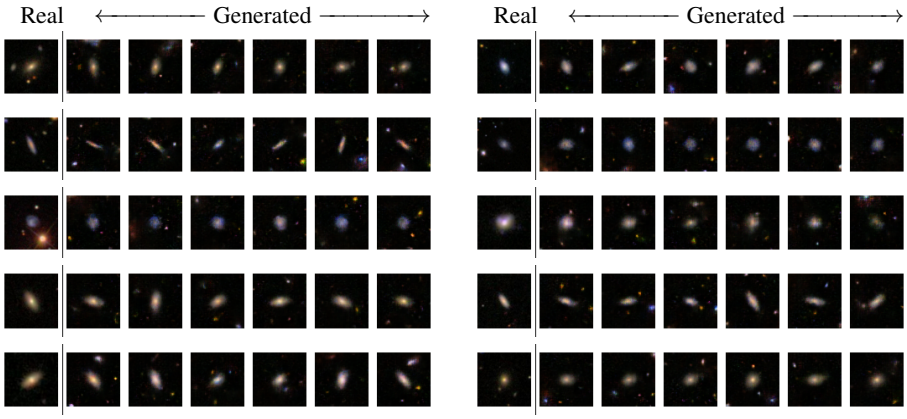

Figure 11: Two sets of samples from the Galaxy Zoo dataset and generated samples using our GAN. Samples in the left columns are real images from the true dataset. For the generated samples, each row uses the same y value (attribute vector) as the true image in the left column, and each column uses a different z value. The attributes y used to generate were not available during training, and held out in a separate test set.

Table 2: RMSE of Galaxy Zoo

Network	Dataset	Data	RMSE
Inception Resnet v2	Galaxy Zoo	Real	0.2126
Inception Resnet v2	Galaxy Zoo	Generated	0.2200
Inception Resnet v2	Galaxy Zoo	Both	0.2152
Alexnet	Galaxy Zoo	Real	0.1697
Alexnet	Galaxy Zoo	Generated	0.1775
Alexnet	Galaxy Zoo	Both	0.1750

Image Quality Assessment Determining the quality of images generated by a GAN is difficult due to the lack of an explicit objective function. We explore various ways of quantitatively and qualitatively analyzing our generated images. We perform a qualitative comparison by generating samples from the Galaxy Zoo and EFIGI datasets and conditioning on attributes from our test set. Figure 11 shows that conditioned on a real attribute, our model is able to generate different novel galaxies which all exhibit the same visual morphology related to that attribute.

Predicting Morphologies We trained two popular networks to predict the galaxy morphologies given an image to assess how accurately our generator was capturing the attributes. For both datasets, we trained Alexnet [13] and Inception Resnet [26] on real data, generated data, and a combination of the two. To evaluate, we calculate the root mean squared error over our holdout test sets. For both datasets, we found Alexnet to outperform Inception Resnet.

Table 3 shows the RMSE results for the EFIGI dataset. While training on the generated data performs slightly worse on the test set, it is a very small amount. Interestingly, using both the generated and real data shows to outperform using only the real data. Table 2 shows the RMSE results for the Galaxy Zoo dataset. Again, training on the real data outperforms the generated data, but only by a small margin, showing that our generated galaxies exhibit accurate morphological features.

Table 3: RMSE of EFIGI

Network	Dataset	Data	RMSE
Inception Resnet v2	EFIGI	Real	0.4042
Inception Resnet v2	EFIGI	Generated	0.4065
Inception Resnet v2	EFIGI	Both	0.4072
Alexnet	EFIGI	Real	0.3703
Alexnet	EFIGI	Generated	0.3875
Alexnet	EFIGI	Both	0.3681

Figure 12: Interpolation samples from the EFIGI dataset. Each row uses a constant z value, but a different y value. First row: interpolation along the arm curvature attribute. Second row: Interpolation along the arm strength attribute. Third row: Interpolation along the dust attribute.

Future work includes training on a larger dataset, allowing the model to be more flexible in the morphologies that it captures. Futhermore, as these images are quite small (64×64), we aim to increase the resolution for a more practical impact.

7 DGAN

This section presents some initial findings on our inprogress work on self driving cars using GANs. Rather than take a reinforcement learning approach, we structure the problem without a specific goal in mind: train an agent to drive a car in simulation such that it is indistinguishable from a human player. This leads away from a discriminative approach and more towards a generative model able to capture a distribution of realistic actions.

Simulation We use the video game Grand Theft Auto V (GTAV) for our simulation environment. With realistic graphics, modifications (mods) able to change the weather, hundreds of diffirent types of cars and motorcycles, and a multitude of landscapes including desert, city, and rural, it is the perfect sandbox environment for teaching an agent. While just a simulation, the work of [24] showed that they were able to improve the realism of generated images by introducing an "enhancer" network. This work could be applied here in order to improve the already very realistic graphics for training on close to real-world data.

Capturing Data We capture data by recording a human play the game, driving around the map with no end location or goal in mind. For each frame, we capture the keys pressed during that frame. Possible actions are: W, A, S, D, WA, WD, DS, DA, NO_KEY, leaving us with a one-hot action vector $y \in \mathbb{R}^9$ (WASD correspond to arrow keys on a keyboard with W=up, S=down, etc.). An in game day takes 48 minutes in real time, meaning that by having a person play the game for an hour we can obtain data across varying lighting conditions. Changing the weather to rainy or cloudy, as well as driving the many different types of cars available can expand our dataset even further [1].

Approach While this project is early in implementation, we have been training a vanilla conditional GAN (not WGAN like the rest of this reoprt) to generate an action given a series of $n = 4$ frames. The viewpoint given to the network is a 3rd person view of the vehicle, as seen in Figure 13.

Figure 13: Two screenshots from GTAV showing very different environment, weather, and lighting conditions. The networks are trained using this viewpoint as input.

While this viewpoint can be changed, it is more natural for a human player and therefore a good first step. The network architecture for the generator and discriminator are the same, and have the same design as the discriminator as seen in [20]. We currently do not have any preliminary results.

8 Conclusion

This report presented three applications of Generative Adversarial Networks for both image generation and image-to-image translation. Qualitative, as well as quantitative results were shown. Our current and future work explores the following:

- UGAN: extend it in order to be end-to-end trainable.
- UGAN: improve the distorter network to capture a wider variety of underwater scenes.
- StereoGAN: Compute disparity using 2+ images to compare with state of the art.
- AstroGAN: Improve the sharpness and quality, as well as extend to further datasets.
- DGAN: Incorporate goals such as a destination to test the generalization of the network.

[1] A full list of vehicles, planes, bikes, and boats that are available in-game (without mods) can be found here: http://grandtheftauto.net/gta5/vehicles

References

[1] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[3] Anthony Baillard, Emmanuel Bertin, Valérie De Lapparent, Pascal Fouqué, Stéphane Arnouts, Yannick Mellier, Roser Pelló, J-F Leborgne, Philippe Prugniel, Dmitry Makarov, et al. The efigi catalogue of 4458 nearby galaxies with detailed morphology. *Astronomy & Astrophysics*, 532:A74, 2011.

[4] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

[5] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[6] Ravi Garg, Vijay Kumar BG, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016.

[7] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, volume 2, page 7, 2017.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[9] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[11] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

[12] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. *CoRR, vol. abs/1703.04309*, 2017.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[15] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conference on Computer Vision*, pages 702–716. Springer, 2016.

[16] Simon Lilly, David Schade, Richard Ellis, Olivier Le Fevre, Jarle Brinchmann, Laurence Tresse, Roberto Abraham, Francois Hammer, David Crampton, Matthew Colless, et al. Hubble space telescope imaging of the cfrs and ldss redshift surveys. ii. structural parameters and the evolution of disk galaxies to z˜ 11. *The Astrophysical Journal*, 500(1):75, 1998.

[17] Chris J Lintott, Kevin Schawinski, Anže Slosar, Kate Land, Steven Bamford, Daniel Thomas, M Jordan Raddick, Robert C Nichol, Alex Szalay, Dan Andreescu, et al. Galaxy zoo: morphologies derived from visual inspection of galaxies from the sloan digital sky survey. *Monthly Notices of the Royal Astronomical Society*, 389(3):1179–1189, 2008.

[18] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[19] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[20] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[21] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[22] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[23] Yoav Y Schechner and Nir Karpel. Clear underwater vision. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2003.

[24] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. Learning from simulated and unsupervised images through adversarial training. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 3, page 6, 2017.

[25] Mike Smith, Ian Baldwin, Winston Churchill, Rohan Paul, and Paul Newman. The new college vision and laser data set. *The International Journal of Robotics Research*, 28(5):595–599, 2009.

[26] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.

[27] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[28] Kyle W Willett, Chris J Lintott, Steven P Bamford, Karen L Masters, Brooke D Simmons, Kevin RV Casteels, Edward M Edmondson, Lucy F Fortson, Sugata Kaviraj, William C Keel, et al. Galaxy zoo 2: detailed morphological classifications for 304 122 galaxies from the sloan digital sky survey. *Monthly Notices of the Royal Astronomical Society*, 435(4):2835–2860, 2013.

[29] Junyuan Xie, Ross Girshick, and Ali Farhadi. Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. In *European Conference on Computer Vision*, pages 842–857. Springer, 2016.

[30] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666. Springer, 2016.

[31] Yiran Zhong, Yuchao Dai, and Hongdong Li. Self-supervised learning for stereo matching with self-improving ability. *arXiv preprint arXiv:1709.00930*, 2017.

[32] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.