Generative Models and Applications in Style Transfer with Interaction

Neil Chandra Arun Drelich Katie Scholl Mitchell Wortsman
Brown University
Providence, RI

Abstract

Recent advances in deep learning have furthered our understanding of image style. Equipped with new technology, we can transfer images across domains and create novel images of a given style. In this work, we survey a variety of generative models and discuss their applications to the problem of style transfer. Moreover, we present an interactive framework for understanding generative models.

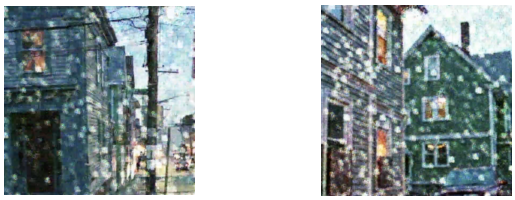

Figure 1. Providence in the style of Monet

1. Introduction

The characterization or transfer of style has long been considered a challenging problem. However, modern generative models have pushed the boundaries of what was previously thought possible.

In this paper, we aim to understand modern generative models and explore their applications in style transfer. Additionally, we develop an interactive framework to visualize the latent space and explore further methods for interaction.

This work is organized as follows. In Section 2, we survey the field of deep generative models and discuss their applications to style transfer. Specifically, we focus on variational auto-encoders (VAEs) and generative adversarial networks (GANs). In Section 3, we present our own methods and experiment with generative frameworks. Finally, we discuss opportunities for interaction in Section 4.

2. Related Work

On Unifying Deep Generative Models

In recent years, remarkable results have been achieved with VAEs and GANs. In [3], Hu et al. show that these two paradigms can be unified under the larger framework of generative models. As such, we begin our survey of related work with an overview of modern generative learning.

Images are high dimensional with complex structure. Therefore, a distribution $p(\mathbf{x})$ over natural images is intractable. Modern generative models consider instead a more tractable latent variable $\mathbf{z}$. An image $\mathbf{x}$ is then said to be generated by some $\mathbf{z}$, and $\mathbf{z}$ may be considered the low dimensional embedding of $\mathbf{x}$.

In this powerful approach, the prior $p(\mathbf{z})$ and likelihood $p(\mathbf{x}|\mathbf{z})$ are both easy to sample from. For example, $p(\mathbf{z})$ is often a multivariate normal with mean 0 and co-variance matrix I. Accordingly, one may sample $\mathbf{x}$ by first sampling $\mathbf{z}$ from $p(\mathbf{z})$ followed by a sample $\mathbf{x}$ from $p(\mathbf{x}|\mathbf{z})$.

Variational Auto-Encoders

Introduced by Kingma and Welling in [7], a variational auto-encoder is a deep learning approach to variational inference. A so-called variational distribution $q(\mathbf{z}|\mathbf{x})$ is learned as an approximation for $p(\mathbf{z}|\mathbf{x})$. The true posterior

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x})p(\mathbf{x}|\mathbf{z})}{\int_{\mathbf{z}'} p(\mathbf{x})p(\mathbf{x}|\mathbf{z}')} \tag{1}$$

is often out of reach as the integral which appears in the denominator is intractable. The defining contribution of the VAE is to exploit the representational capacity of deep neural networks to approximate both the variational distribution and likelihood. Figure 2 shows a VAE, with trapezoids denoting deep neural networks.

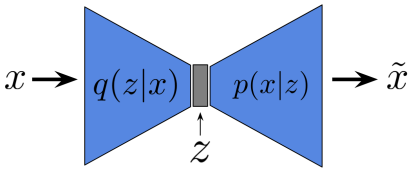

Figure 2. A Variational Auto-Encoder

In practice, the likelihood is parameterized by θ and the variational distribution is parameterized by ϕ. These parameters θ, ϕ are tuned so that the output $\tilde{\mathbf{x}}$ resembles the input $\mathbf{x}$. Intuitively, $\mathbf{z}$ may then be interpreted as a low dimensional embedding of $\mathbf{x}$. The loss is then a function of θ, ϕ as follows.

$$\begin{aligned}\mathcal{L}_{\theta,\phi} =& \mathbb{E}_{\mathbf{z}\sim q_\phi}\left[-\log p_\theta(\mathbf{x}|\mathbf{z})\right] \\ &+ \mathcal{D}_{\mathrm{KL}}\left(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z})\right)\end{aligned} \tag{2}$$

The cross-entropy term $\mathbb{E}_{\mathbf{z}\sim q_\phi}\left[-\log p_\theta(\mathbf{x}|\mathbf{z})\right]$ may be interpreted as a reconstruction loss while the KL-Divergence $\mathcal{D}_{\mathrm{KL}}\left(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(z)\right)$ imposes the desired prior on $\mathbf{z}$.

A prior $p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, \mathbf{I}_k)$ is often used for the k-dimensional latent vector $\mathbf{z}$. Here, $\mathcal{N}$ denotes the pdf of the multivariate normal and $\mathbf{I}_k$ is the $k \times k$ identity matrix. The variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$ is then $\mathcal{N}(\mathbf{z}; \mu_\phi(\mathbf{x}), \mathbf{I}_k * \sigma^2_\phi(\mathbf{x}))$, where $\mu_\phi(\mathbf{x})$ and $\sigma^2_\phi(\mathbf{x})$ are implemented via a neural network.

Conditional Variational Auto-Encoders

Kingma et al. proposed a new approach to semi-supervised learning with deep generative models in [6]. In particular, the "generative semi-supervised model" establishes a useful framework for conditioning generation on a particular latent variable. The conditional VAE (CVAE) operates under the following model of the data: Consider a set of observations and corresponding class labels $\{(\mathbf{x}_i, y_i)\}, i \in \{1, \ldots, N\}$ where $\mathbf{x}_i \in \mathbb{R}^d$ and discrete $y_i \in \{1, \ldots, L\}$. We assume that observation $\mathbf{x}$ is generated by both a latent code $\mathbf{z}$ as well as a latent class code y, with a categorical prior on y in addition to a Gaussian prior on $\mathbf{z}$.

As a result, we describe the variational distribution $q_\phi(\mathbf{z}|\mathbf{x}, y) = \mathcal{N}(z; \mu_\phi(\mathbf{x}, y), I_k * \sigma^2_\phi(\mathbf{x}, y))$. An additional variational approximate posterior is introduced through the classifier, $q_\phi(y|\mathbf{x}) = \mathrm{Cat}(y; \pi_\phi(\mathbf{x}))$. The likelihood $p_\theta(\mathbf{x}|\mathbf{z}, y)$ is accordingly conditioned on both $\mathbf{z}$ and y. As before, μ_ϕ, σ^2_ϕ, and π_ϕ are represented by neural networks.

Learning the parameters θ, ϕ is accomplished by maximizing the variational lower bound on the marginal likelihood.

Generative Adversarial Networks

Generative adversarial networks have been integral to the recent success of deep learning. Proposed by Goodfellow et al. in [2], they have since found a variety of applications. GANs use a minimax framework consisting of a Generator G and Discriminator D. The interaction between G and D is illustrated by Figure 3.

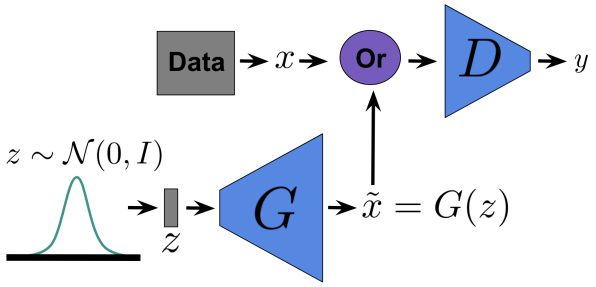

Figure 3. A Generative Adversarial Network

The generator is given noise and aims to produce images which appear as though they are drawn from the dataset. Alternatively, the discriminator's goal is to determine the probability that a given image is included in the dataset. The discriminator is either given a real image x from the dataset or the generated image $\tilde{x} = G(z)$. The loss is then

$$\begin{aligned}\mathcal{L}(G, D) =& \min_G \max_D \; \mathbb{E}_{x\sim p_{data}}[\log D(x)] \\ &+ \mathbb{E}_{z\sim p_z}[\log\left(1 - D(G(z))\right)]\end{aligned} \tag{3}$$

where $\mathbb{E}_{x\sim p_{data}}[\log D(x)]$ pushes the discriminator to correctly identify images from the dataset. Alternatively, $\mathbb{E}_{z\sim p_z}[\log\left(1 - D(G(z))\right)]$ encourages the generator to create realistic fake images while incentivizing the discriminator to identify these images as fake.

Neural Style Transfer

Gatys et al. utilize a fully convolutional approach to style transfer in [1]. They consider the problem of taking two images, p and a, and outputting x with the content of p but the style of artwork a. The defining contribution of the paper is that Convolutional Neural Networks (CNNs) are capable of separating the paper's representations of style and content. In their 19-layer VGG-Network, they encode a representation of content from filter responses to p at each layer of the network. The style representation is defined as the response of a different filter bank to a. They minimize the following loss term to produce the output image x with the style of a and content of p, where α and β are constants.

$$\mathcal{L}_{total}(p, a, x) = \alpha\mathcal{L}_{content}(p, x) + \beta\mathcal{L}_{style}(a, x) \tag{4}$$

Here, $\mathcal{L}_{content}(p, x)$ measures the distance between the content representations of p and x while $\mathcal{L}_{style}(a, x)$ measures the distance between style representations of a and

x. The gradient is then computed with respect to x and an output is produced that jointly minimizes content loss and style loss. The approach iteratively produces a 'better' x for each input pair (p, a). Our implementation of this net took roughly a minute to compute x on a CPU with inputs of size 444x444 and an output of size 128x128.

Figure 4. An individual in Picasso's style using the method of [1].

Pix2Pix

Isola et al. use a GAN for image-to-image translation in [4]. They consider image pairs from two data sets A and B. For each image pair (z, x), $x \in B$ is the translation of $z \in A$ across domains. Their system, as illustrated by Figure 5, is then very similar to a traditional GAN. However, the discriminator now observes image pairs of the form (z, x) or $(z, \tilde{x})$. The discriminator aims to classify pairs (z, x) as real and pairs $(z, \tilde{x})$ as fake. Here, $\tilde{x} = G(z)$ as defined below.

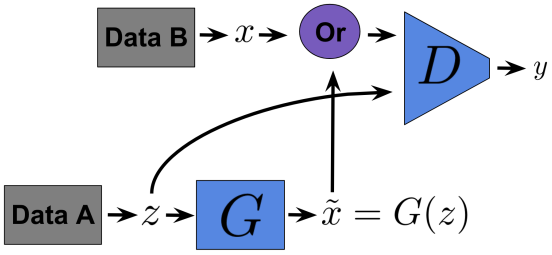

Figure 5. Pix2Pix

In addition to the traditional GAN loss, Pix2Pix introduces an L1 loss $\|G(z) - x\|_1$. Isola et al. achieve impressive results translating facades to buildings, day to night, edges to photos, and more. However, they require that their data consist of image pairs, which may be limiting.

CycleGAN

In [10], the authors present a mechanism for translating images across domains. Unlike Pix2Pix, they do not require that the dataset consist of image pairs. Instead, they need only a set of images from domain A and a set of images from domain B. The authors then have two GANs: (G_A, D_A) maps from domain A to B and (G_B, D_B) maps in the opposite direction. In addition to the GAN loss found in equation 3, the authors include a cycle-consistency loss $\mathcal{L}_{\text{cycle}}$. This loss ensures that $G_A(G_B(x)) \approx x$ and $G_B(G_A(y)) \approx y$. Formally, this cycle consistency loss is given as

$$\begin{aligned}\mathcal{L}_{\text{cycle}}(G_A, G_B) =& \mathbb{E}_{x\sim p_A}[\|G_B(G_A(a)) - x\|_1] \\ &+ \mathbb{E}_{y\sim p_B}[\|G_A(G_B(y)) - y\|_1].\end{aligned} \tag{5}$$

3. Methods and Results

In this section, we detail our exploration and experimentation with generative models. We use the PyTorch deep learning framework to create our own implementations. The code may be found at `https://github.com/mwortsma/elements_of_style`.

3.1. MNIST VAE

We begin by implementing a vanilla VAE on the MNIST dataset. MNIST consists of a training set of 60000 handwritten digits, each formatted as a 28×28 grayscale image.

We use fully-connected units with ReLU activations. The encoder $q_\phi(\mathbf{z}|\mathbf{x})$ and decoder $p_\theta(\mathbf{x}|\mathbf{z})$ consist of one hidden layer each. Despite the simplicity of the network, we obtain meaningful results.

As illustrated by Figure 6, the network is capable of reconstructing images. On the left is a random subset $\mathbf{x}$ of the dataset. We then sample $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$ and reconstruct $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$ on the right.

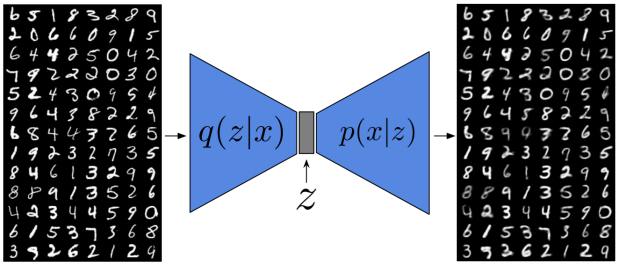

Figure 6. Reconstructions using a 10-dimensional latent $\mathbf{z}$.

Recall that our generative framework should allow us to construct novel images by sampling from $p_\theta(\mathbf{x})$. We may do so by first sampling $\mathbf{z}$ from $\mathcal{N}(\mathbf{z}; 0, I_k)$ followed by sampling $\mathbf{x}$ from $p_\theta(\mathbf{x}|\mathbf{z})$. As illustrated by Figure 7, this process creates semi-realistic digits.

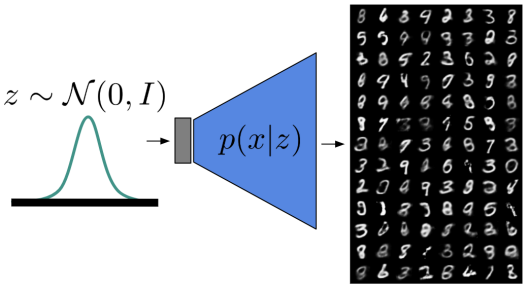

Figure 7. Samples from $p_\theta(\mathbf{x})$ using a 10-dimensional latent $\mathbf{z}$.

Finally, we aim to visualize the latent space. We begin by considering the case where $\mathbf{z}$ is a 2-dimensional vector. In Section 4.1, we use interaction to extend our visualizations to higher dimensions. We also experiment with using PCA to visualize higher dimensions but find that it is unsuccessful. This is likely due to the imposed prior on $\mathbf{z}$.

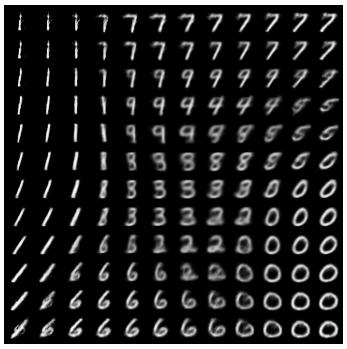

Figure 8. Visualizing a 2-dimensional latent space.

In Figure 8, each axis corresponds to a dimension of $\mathbf{z}$. At each point (z_1, z_2) we show a sample from $p_\theta(\mathbf{x}|(z_1, z_2))$. This is similar to the manifold visualization of [7].

Next we consider a subset of the dataset and place the corresponding 2-dimensional embedding vectors on a Cartesian plane. We show the results in Figure 9. Each point is also color-coded to correspond with its associated class label and we place some corresponding images on the plane. It is interesting to note that we observe some clusters which correspond to class labels. This suggests that we may learn a lot about the data from the embedding space.

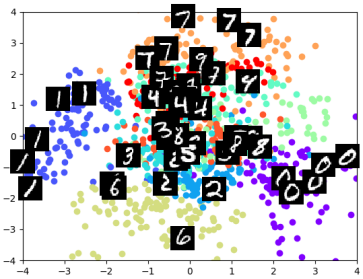

Figure 9. Placing 2-dimensional embeddings on a Cartesian plane.

3.2. MNIST CVAE

As illustrated by Figure 9, the latent space contains information about both style (i.e. handwriting) and content (the digits themselves). In order to better separate out these two characteristics, we implement the aforementioned CVAE with some notable differences.

The probabilistic encoder $q_\phi(\mathbf{z}|\mathbf{x}, y)$, classifier $q_\phi(y|\mathbf{x})$, and decoder $p_\theta(\mathbf{x}|\mathbf{z}, y)$ are implemented using MLPs with 500 hidden units each and Softplus activations, as in [6]. In order to ensure valid parameterization for $\text{Cat}(y; \pi_\phi(\mathbf{x}))$, we transform the output of the classifier using the softmax function.

We had initially conditioned the encoder only on $\mathbf{x}$, but found that conditioning on both $\mathbf{x}$ and y significantly improved results when constructing style analogies, as shown in Figure 10.

Rather than implement the semi-supervised model described in the paper, we focus on the supervised case, as this was most relevant to the problem of style transfer. Our goal is primarily to separate content from style in the latent encoding of an image. MNIST provides a concrete notion of content with class labels based on digits.

Figure 10. MNIST analogies, conditioning the encoder either only on $\mathbf{x}$ (left) or on both $\mathbf{x}$ and y (right). For an input image $\mathbf{x}$ (left column), we run inference to obtain latent "style" vector $\mathbf{z}$ and vary latent y across digit classes 0-9 (right columns).

3.2.1 Convolutional Architecture

In addition to the fully-connected approach, we follow [5] and implement a convolution-based CVAE. We introduce $\varphi_z(\mathbf{x})$ and $\varphi_y(\mathbf{x})$, which learn to extract salient features from an image. Each feature is used in lieu of $\mathbf{x}$ as input for the MLPs. We can now describe the variational distribution in terms of φ_z and φ_y:

$$\begin{aligned} q_\phi(\mathbf{z}|\mathbf{x}) &= \mathcal{N}(\mathbf{z}; \mu_\phi(\varphi_z(\mathbf{x})), I_k * \sigma^2_\phi(\varphi_z(\mathbf{x}))) \\ q_\phi(y|\mathbf{x}) &= \text{Cat}(y; \pi_\phi(\varphi_y(\mathbf{x}))) \end{aligned}$$

The probabilistic decoder $p_\theta(\mathbf{x}|\mathbf{z}, y)$ is replaced by a deep convolutional network. The network learns to upsample a joint latent feature $\psi_\theta(\mathbf{z}, y)$ via groups of transposed convolutions, batch normalization, and ReLU activation. We represent ψ_θ as an MLP with 256 hidden units.

The entire convolutional CVAE is trained end-to-end.

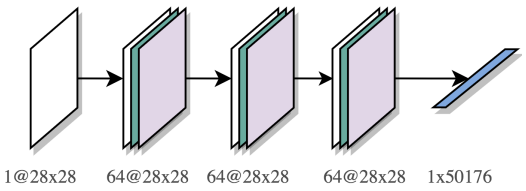

Figure 11. Convolutional architecture for $\varphi_z(\mathbf{x})$, which is used in the encoder. For an input image, successive convolution, batch normalisation and ReLU activation is applied at each layer. The final feature is a flattened 50176-dimensional vector.

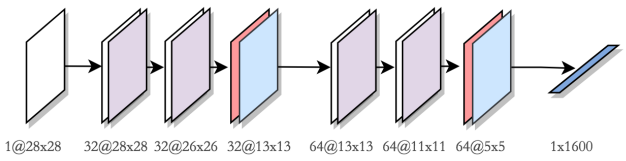

Figure 12. Convolutional architecture for $\varphi_y(\mathbf{x})$, which is used in the classifier. For an input image, successive convolution, ReLU activation, max pooling, and dropout is applied, in that order. The final feature is a flattened 1600-dimensional vector.

We compare the effectiveness of the convolutional approach in its ability to construct analogies across MNIST digit classes. As illustrated by Figure 16, the results display a more consistent style across classes when compared to Figure 10. In addition, analogies generated from the convolutional CVAE suffer from less ambiguity of content.

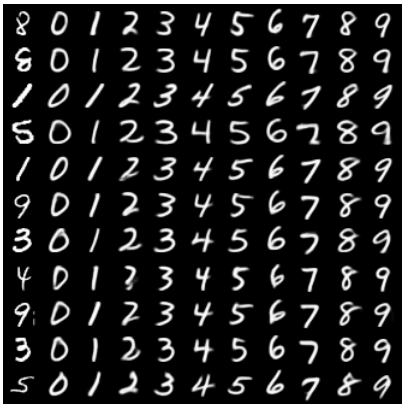

Figure 13. MNIST analogies constructed with a conditional VAE.

3.3. Cross-Dataset Analogies using CVAEs

We show a potential application of CVAEs for transferring style across datasets. First, we learn a convolutional CVAE on MNIST digits. We then run inference on examples drawn from FashionMNIST [9], a dataset with identical image size and structure to MNIST. Analogies are constructed as before, by sampling $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$ and varying y across each of the 10 class labels.

Although this is not a well-defined problem, we observe digit reconstructions that match the general "shape" of the input image. We consider this a positive result, and an interesting method that could be explored further in future work.

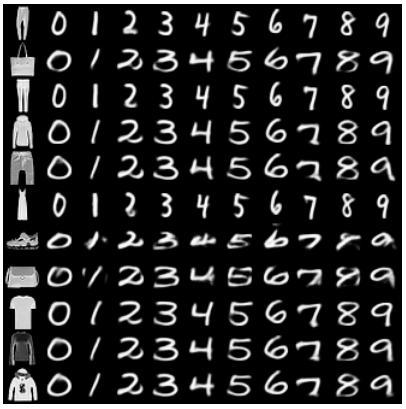

Figure 14. MNIST analogies of FashionMNIST images. Note how the shape of the shoe is reflected in its digit analogies.

3.4. Variational Transfer Encoder

In line with the idea of cycle consistent loss in [10], we investigate the idea of training a VAE on a dataset of image pairs with identical content but different styles. At a high level, the approach in [10] simultaneously trains two VAEs that have shared generators G_A and G_B such that for an image x in image domain B, $x \approx G_A(G_B(x))$, and vice versa. The intent of our approach is to train the encoder to strip input images of the source style and the decoder to

reinsert the target style into the underlying content vector.

We implement a VAE with an encoder containing four convolutional layers and two fully-connected layers and a decoder containing two fully-connected layers and four deconvolutional layers. The purpose of this net is to transfer the style of the target image to the source image and is correspondingly labeled a Variational Transfer Encoder (VTE).

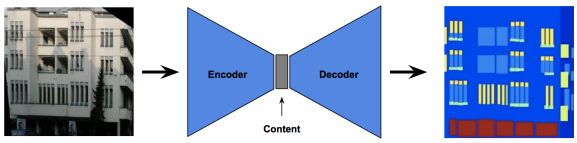

Figure 15. Proposed VTE model

The *facades* dataset of Pix2Pix provides image pairs of differing style but identical content. Loss is computed by summing the MSE and KL-Divergence of the VTE output and the target image. With this architecture and dataset, loss plateaus at a high value after only 50 epochs. This result suggests that without the adversarial loss term present in [10] the gradient is hard to descend. Additionally, the net does not enforce that the intermediary vector actually contains a content representation of the input image.

4. Opportunities for Interactions

4.1. Exploring the Latent Space

Recall that the learned embeddings $\mathbf{z}$ from our VAEs are distributed $\mathcal{N}(0, I_k)$. An effective embedding representation would return reasonable decoded results for values sampled close to this mean. In order to understand how effectively the model represents the data, we implement a tool to visually explore the space around the mean of this distribution.

We present an interactive tool with sliders for each dimension of $\mathbf{z}$ which allows the user to modify the values of the latent code directly. As the sliders are changed, the images displayed along the top of the widget update in real-time to match the newly-decoded latent $\mathbf{z}$ vector. When using a vanilla VAE, where reconstructions are not explicitly conditioned on class labels, the widget displays a single image. Under the CVAE approach, the widget displays a decoded image for each possible class label, presenting all possible analogies for the selected $\mathbf{z}$.

The widget provides a wealth of qualitative information about the latent space. In general, we find that different dimensions encode distinct stylistic qualities of the resulting image, such as slant, boldness, curvature, or width. Exploring the latent space with this tool provides meaningful insight. The widget may aid in indicating when we have too many or too few latent dimensions.

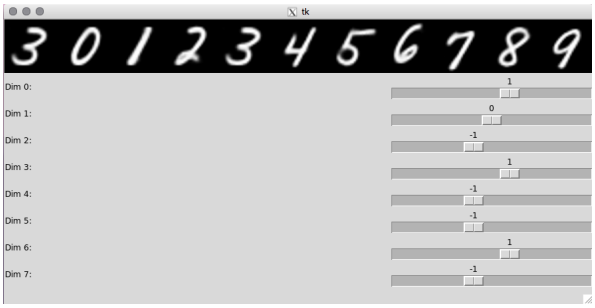

Figure 16. The slider-based widget

4.2. Music Videos

Finally, we explore interactions between style transfer and music. We believe that this technique can be used in movies and music videos. Although we do not achieve success, we hope that our exploration encourages future work.

We train CycleGAN on Monet2Photo as in [10]. We then experiment with a variety of techniques to couple style transfer and music. For each technique, we consider an input video of frames $i_1, ..., i_n$. First, we construct the output frame o_k via a linear combination of the input frame i_k and input frame with style transfer applied. We let F be the function which applies style transfer. In the case of CycleGAN's Monet2Photo, this is the generator G_B. We let

$$o_k = (1 - d_k) * i_k + d_k * F(i_k) \tag{6}$$

where $d_k \in [0, 1]$ measures how close frame i_k is to a beat. We compute d_k by dividing the difference of the next beat and the current time by the period of the music. The python library Librosa is used to compute the beats. As seen in `elements_of_style/monet2photo.mp4`, this produces the interesting effect of bursts of style occurring alongside each beat. We use the song *Pay No Mind* off the new album by Beach House.

Additionally, we experiment by adding Gaussian or uniform noise to our image before we apply style transfer. We apply this noise as a function of i_k's closeness to a beat. However, we may also apply this noise as a function of the amplitude of the sound wave.

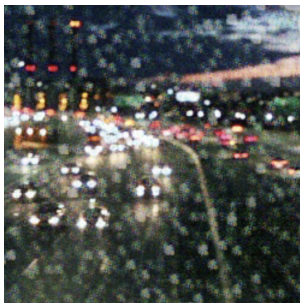

Figure 17. Interstate 195 in the style of Monet

As illustrated by Figures 1 and 17, each individual frame appears in the style of Monet. However, we are unsuccessful as there is no continuity between frames. The authors of [8] present a remedy for this, which we will look to include in our future work.

5. Conclusions and Future Work

We believe that a better understanding of generative models will follow from the ability to explore and visualize the latent space. Our interactive tool is a step in this direction.

Moreover, we believe that interaction paired with style transfer could produce a number of interesting effects. Music is only the beginning. With the growing role of virtual reality, humans could interact with a style-transferred environment in VR. Notable challenges remain in making real-time style transfer more effective, broadening its applications and pushing beyond restricted domains.

References

[1] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv*, Aug 2015.

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[3] Z. Hu, Z. Yang, R. Salakhutdinov, and E. P. Xing. On unifying deep generative models, 2017.

[4] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. 2016.

[5] B. Keng. Semi-supervised learning with variational autoencoders. `http://bjlkeng.github.io/2017/`. Accessed: 2018-05-16.

[6] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling. Semi-supervised learning with deep generative models, 2014.

[7] D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2013.

[8] M. Ruder, A. Dosovitskiy, and T. Brox. Artistic style transfer for videos. 2016.

[9] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[10] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2017.