CATlation: Style Transfer with Diffusion

Nicky Kriplani
New York University
ak9100@nyu.edu

Pratyush Avi
New York University
pa2439@nyu.edu

Anna Choromanska
New York University
achoroma@gmail.com

Abstract

Unsupervised Image-to-Image Translation aims to translate style characteristics between visually different images. Although there are plenty of existing methods to solve this problem, most of them are extremely complex and/or use GANs, which have been superseded by diffusion models. We adapt an existing technique that uses GANs, aiming to simplify this model with the generative power of diffusion models. We also look to transfer this task to the music translation case, in which there is little to no literature using diffusion models. All of our code is available at `https://github.com/NickyDCFP/Style-Transfer-with-Diffusion/`. Music samples are available at `https://www.youtube.com/watch?v=kQ7oCG1an10`.

1 Introduction

Image generation models have received a large amount of attention due to their ability to produce remarkably photorealistic images. Specifically, these models are renowned for their proficiency in conditional generation, a context where a user inputs a conditional parameter that affects the features in the final outputted image. For example, text-conditioned image generation allows users to input a textual prompt with a description of the image, and the output should match that description. Image-to-image translation slightly modifies this task by "translating" style characteristics from a style image to a desired input image. A sample of image-to-image translation is available in Figure 1. Overall, these techniques allow for a large degree of control in the output using conditional information passed to the model.

For a while, GANs[5] were the most prominent and successful class of models for these tasks. Recently, however, research into diffusion models has seen them supersede GANs for text-conditioned image generation and image-to-image translation tasks [20, 22, 23]. Our work hopes to expand on these developments in the Unsupervised Image-to-Image Translation context, specifically hoping to find a simpler solution than the existing literature, which tends to be relatively complex (i.e, using separate advanced techniques like contrastive learning with patching [12]).

Our Contributions We propose a new technique for image-to-image style transfer using diffusion models. With no pretraining and minimal added complexity, our model operates by simply altering the loss function and modifying the diffusion noise schedule. Overall, we adapt and simplify existing methods commonly used in GANs to diffusion models for the task of Unsupervised Image-to-Image Translation.

Our work also extends the style transfer task to music. Music generation with diffusion models has seen increased interest lately, with works like [1, 8] defining the state of the art with diffusion models. Music translation, however, has been incredibly sparsely addressed and, to our knowledge, has not been done at all with diffusion models. Therefore, we present the first foray into music style transfer using diffusion models as well.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

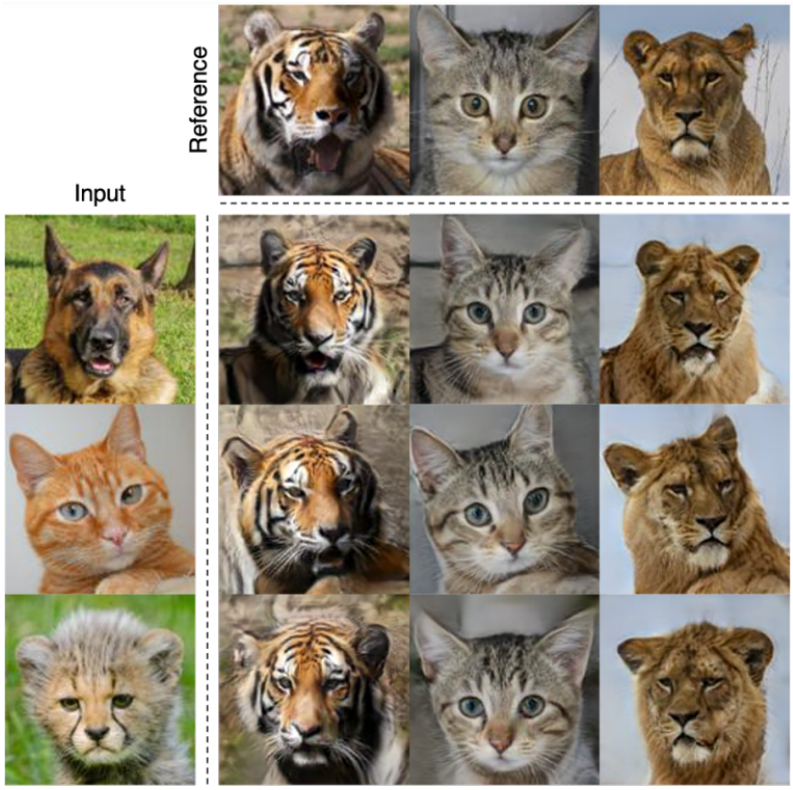

Figure 1: A sample of the Unsupervised Image-to-Image translation task, from [12]. Crucially, we use slightly different terminology than [12]. We denote their "input" and "reference" images as the "style" and "input" images, respectively, which we find to be more intuitive than the original names.

2 Background

Unsupervised Image-to-Image Translation The task of Unsupervised Image-to-Image Translation seeks to learn a joint distribution of paired images from two domains [14]. Consider two domains $\mathcal{X}_1$ and $\mathcal{X}_2$. Given two marginal distributions $p_{\mathcal{X}_1}(\mathbf{x}_1)$ and $p_{\mathcal{X}_2}(\mathbf{x}_2)$ on these domains, an infinite number of joint distributions can arise. While in the supervised domain we would be given pairs of images to learn this distribution, the unsupervised task does not offer this luxury.

Instead, we operate under a shared latent space assumption, which helps us to learn the pairs. Specifically, we assume that for any set of paired images $(\mathbf{x}_1, \mathbf{x}_2)$ in our desired joint distribution on $\mathcal{X}_1 \times \mathcal{X}_2$, there exists a point $\mathbf{z}$ in some shared latent space $\mathcal{Z}$ that can reversibly and uniquely map to both images. That is, there exist functions E_1, E_2, G_1, and G_2 such that $E_1(\mathbf{x}_1) = E_2(\mathbf{x}_2) = \mathbf{z}$ and that $\mathbf{x}_1 = G_1(\mathbf{z})$, and $\mathbf{x}_2 = G_2(\mathbf{z})$, where $\mathbf{z}$ is the corresponding latent code for any given pair $(\mathbf{x}_1, \mathbf{x}_2)$ in our target distribution. This relationship is illustrated in Figure 2.

This way, we can learn the functions $F_{1\to2}$ and $F_{2\to1}$, where $F_{1\to2}(\mathbf{x}_1) = G_2(E_1(\mathbf{x}_1)) = \mathbf{x}_2$ and $F_{2\to1}(\mathbf{x}_2) = G_1(E_2(\mathbf{x}_2)) = \mathbf{x}_1$. Crucially, though, the existence of these functions also gives rise to a cycle consistency constraint. Stated simply, the cycle consistency constraint requires that $F_{1\to2}(F_{2\to1}(\mathbf{x}_1)) = \mathbf{x}_1$ and $F_{2\to1}(F_{1\to2}(\mathbf{x}_2)) = \mathbf{x}_2$. In other words, since the point in the latent space uniquely maps to that pair, translating an image from one domain to the other and then back should yield the same image. Enforcing this constraint will be the core of how we learn the distribution as it requires the selective distinction of recoverable characteristics: during translation, if concepts in the image are changed besides those which can be recovered when translating back, then translating back will be much harder or even impossible.

Denoising Diffusion Probabilistic Models Denoising Diffusion Probabilistic Models, or Diffusion models [7], are a class of generative models that learn to generate random images through a denoising

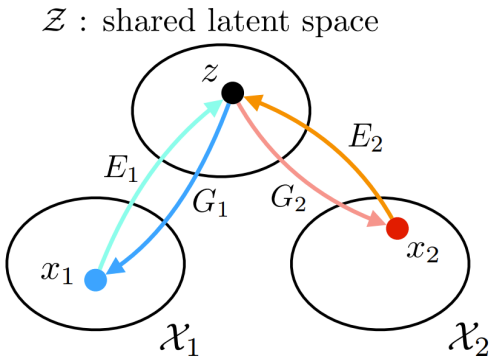

Figure 2: Pulled from [14]. Demonstration of the shared latent space assumption.

process defined by a Markov chain. At training time, an input image $\mathbf{x}_0$ is gradually noised according to a variance schedule until the original signal is destroyed (for T timesteps), yielding a final noise vector $\mathbf{x}_T$. The model learns to invert this forward diffusion process with a reverse process that predicts the added noise at each step.

At inference, one begins with a random Gaussian noise vector $\mathbf{x}_T$. At each timestep t, the model predicts the residual noise $\epsilon_t = \mathbf{x}_{t+1} - \mathbf{x}_t$ between its input and a slightly less noised version. For the predicted ϵ_θ, the subsequent $\hat{\mathbf{x}}_t = \mathbf{x}_{t+1} - \epsilon_\theta$ is passed back into the model for the next denoising step. T passes through the model eventually yield a final vector $\mathbf{x}_0$, which is the generated output.

These models are trained with a basic reconstruction loss on the residual noise, defined as

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{x}_t, t, c, \epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\|\epsilon_t - \epsilon_\theta(\mathbf{x}_t, t, c)\|_2^2 \right]$$

where c is a conditional parameter that allows for guidance in the output.

Log-Mel Spectrograms Raw waveforms can be extremely large in size, with only one second of audio easily requiring upwards of 20,000 samples/entries. For this reason, modern music generation pipelines tend to generate a low-dimensional representation of the final waveform before using a vocoder model to decode the representation into samples [4].

An extremely common choice for this representation is the log-mel spectrogram, which is generated by first creating a spectrogram using the short-time Fourier transform (STFT), then by rescaling the spectrogram to the mel scale, and finally by taking its log. The mel scale is a specific scale for audio spectrograms that spaces pitches such that changes are directly proportional to human pitch perception [16]. For example, the subjective (human-perceived) increase in pitch between $400 \rightarrow 500$ mels and $500 \rightarrow 600$ mels is the same, even if the objective increases in frequencies between those mappings are different. This makes the mel spectrogram more easily human-readable than traditional spectrograms.

Mel spectrograms are also lower dimensional than traditional spectrograms, which reduces memory constraints. Taking the log of a mel spectrogram constrains the values to a smaller range, which makes them more suitable for learning after normalization.

3 Related Work

Image Translation with Diffusion Models Conditional diffusion models allow for customization in image generation tasks. [19] demonstrates this for text-to-image generation by taking an input image and generating new images with the primary subject based on a text prompt. Unlike our work, however, they focus on using text for the style information and perform a supervised learning task that also explicitly encodes a prior preservation loss. [20] present image-to-image translation using diffusion models. Their model, Palette, outperforms GAN models for the same task without any task-specific hyper-parameter tuning or architecture customization, demonstrating that diffusion models are a good fit for image-translation tasks.

[22] and [23] introduced models based on the idea that pretrained diffusion models boost image-to-image translation. Both their models (PITI and ControlNet) trained a task-specific model (for image translation) to work with a pretrained diffusion model (which generated images). The task-specific model would encode the style (or translation input) into a task-agnostic latent space that would be given as an input to the pretrained image generator model. They used canny edge images, hough transforms, segmentation masks, and similar transforms for the task-specific model. However, neither of these models were targeted at pose-transfer in images. Instead, they focused on "painting in" masks, maps, or edge images to generate various samples.

Mel Spectrogram Inversion Because of the usefulness of log-mel spectrograms for audio generation, there has been a decent volume of work dedicated to finding more sophisticated ways to invert these spectrograms into waveforms. [4] presents perhaps the most novel method for log-mel spectrogram inversion in the context of music generation. They also note a key gap in this field for the music generation task: many existing models, such as [10, 11, 15] are specifically designed to invert speech. For this reason, these models can produce artifacts that are not very noticeable in spoken word (such as small variations in pitch or tempo) but that significantly worsen the quality of generated audio. Based on that gap, [4] develops a model specifically tuned for music. Since their model is private, though, we use [11] for inversion, which, to our knowledge, is the most sophisticated public model.

Conditional Music Generation With the success of diffusion models for image generation, there has been a large amount of work attempting to translate that efficacy to the music generation task. Most notably, [1, 8] use text-conditioned diffusion models to generate brief music clips. There has also been an effort to generate music with more unique conditioning mechanisms, such as in [3]. [13] introduces a series of unique tasks such as filling in gaps in tracks and demonstrates how diffusion models can complete these tasks effectively. However, the task of unsupervised translation for music is relatively young. [1] introduced, alongside their high-fidelity text-conditioned diffusion models, a mechanism by which a user could input a whistled melody and use text to describe a style that could be translated onto that melody. The only somewhat recent work we can find, however, that focuses on the unsupervised translation task, is [21], and even this was written before the advent of diffusion models.

4 Methods

Datasets For this task, we chose to operate on the AFHQ [2] dataset of animal faces and GTZAN [6] dataset of music and music genres. We represent independent examples as coming from an unlabeled set of, depending on the dataset, images or tracks $X \in \mathbb{R}^{N \times C \times H \times W}$. For each of these datasets, we represent the classes (which are obscured during training) as domain characteristics that should not be translated.

In the AFHQ case, to translate between two classes, we aim to transfer the class-independent characteristics such as pose from a "style" image $\mathbf{x}_s \in X$ onto that of an "input" image $\mathbf{x}_i \in X$. The input image, in turn, provides the class (animal species) for the generated image.

For GTZAN, we consider a music genre to be a class characteristic, so we aim to transfer class-independent style characteristics from $\mathbf{x}_s$ onto the genre of $\mathbf{x}_i$.

Architecture For our base architecture, we use an Attention Residual U-Net. A general overview of this style of architecture can be found in Figure 3. Our image model had four contracting blocks, each with respective output channels of $64 \to 128 \to 384 \to 1536$. Each contracting block consisted of 3 ResBlocks, and there was a 2×2 MaxPooling layer between successive contracting blocks. There were also 4 attention heads for each attention gate. These attention gates were only present at the first 3 of the 4 expanding blocks. All activations in the model were the Swish/Sigmoid Linear Unit (SiLU) [17] function, which is defined as

$$\text{Swish}(\mathbf{x}) = \mathbf{x}\sigma(\mathbf{x})$$

where $\sigma(\mathbf{x})$ is the sigmoid function. In the image case, our architecture is identical except that we use 7 contracting blocks, each with channel outputs of $64 \to 64 \to 64 \to 128 \to 256 \to 512 \to 1536 \to 6144$.

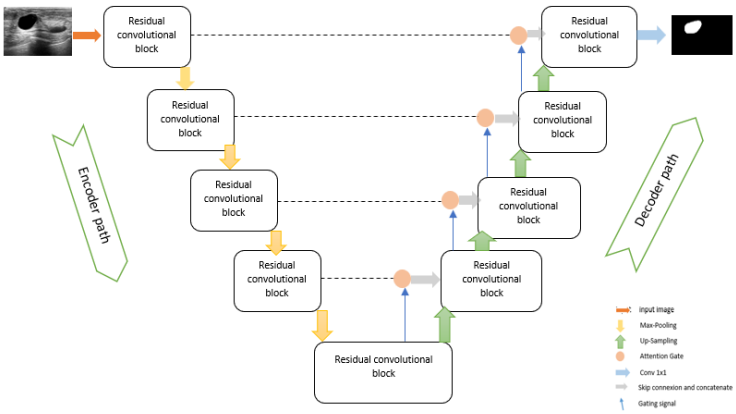

Figure 3: Attention Residual U-Net, from [9]. The encoder path consists of a set of contracting blocks that compress the input down into a latent code. This code is decoded with expanding blocks, each of which receive a residual with attention from their corresponding contracting blocks.

Similar to [12], we use a latent style code $\mathbf{z}_s$ generated by a style encoder $\mathcal{E}$. This style code is passed into the model akin to standard conditional guidance, meaning it is overlaid with a timestep embedding. Our final basic diffusion loss function is

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{\mathbf{x}_i^t, t, \mathbf{z}_s \sim \mathcal{E}(\mathbf{x}_s), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}[\|\epsilon_t - \epsilon_\theta(\mathbf{x}_i^t, t, \mathbf{z}_s)\|_2^2]$$

Our models were trained for 500 epochs on 4 RTX 8000s with a batch size of 64 (16 per GPU) for the image set and 32 (8 per GPU) for the music set.

Cycle Consistency In order to enforce the style transfer, we add a cycle consistency-based regularization term to our loss function in a similar manner to [12]. [12] used, for a given model G, a term of

$$\mathcal{L}_{\text{cyc}} = \mathbb{E}_{\mathbf{x}_i, \mathbf{x}_s, \mathbf{z}_s \sim \mathcal{E}(\mathbf{x}_s)}[\|\mathbf{x}_s - G(\mathbf{x}_i, \mathbf{z}_s)\|_1]$$

For diffusion models, the term must be slightly different as the generator does not directly predict an output image. Instead, we operate on the residual difference between the noised $\mathbf{x}_i^t$ and the style image $\mathbf{x}_s$, which gives rise to

$$\mathcal{L}_{\text{cyc}} = \mathbb{E}_{\mathbf{x}_i^t, t, \mathbf{x}_s, \mathbf{z}_s \sim \mathcal{E}(\mathbf{x}_s)}[\|(\mathbf{x}_i^t - \mathbf{x}_s) - \epsilon_\theta(\mathbf{x}_i^t, t, \mathbf{z}_s)\|_1]$$

Note that $\mathcal{L}_{\text{diff}}$ pushes the output toward $\mathbf{x}_i^{t-1}$, a denoised version of $\mathbf{x}_i^t$. Since $\mathbf{x}_i^t$ is our best approximation for $\mathbf{x}_i^{t-1}$, this loss tends the output of the model towards $\mathbf{x}_s$. Intuitively, just like how an ℓ_1 term encourages sparsity constraint in a regression problem, this term could encourage "relative sparsity" between the output and the style image by encouraging borders and edges (darker/sparser areas) in the output image to arise in similar places to the style image. When dampened with a scaling term λ_{cyc}, this term thus has the effect of molding the edges of the output image closer to the style image. We used λ_{cyc} values of 0.05 for the image case and 0.1 for the music case.

Subsequently, the $\|\cdot\|_2^2$ term that tends towards the input image would encourage the output to be colored in a similar way to said input image, because small differences in pixel values are inflated by the squared term. When tuned properly, this can almost be thought of as a two-step process in which the ℓ_1 term "draws out an outline" of the output image using style/pose characteristics from the style image, and the $\|\cdot\|_2^2$ term colors that image in using the color/class characteristics of the input image.

We can also take an alternative approach. Instead of biasing the current latent towards the style image, we can bias the underlying input image towards the style image. This alternative gives rise to

$$\mathcal{L}_{\text{cyc}} = \mathbb{E}_{\mathbf{x}_i^0, t, \mathbf{x}_s, \mathbf{x}_i^t \sim \mathbf{x}_i^0 + \epsilon_{0:t}^{\text{scaled}}, \mathbf{z}_s \sim \mathcal{E}(\mathbf{x}_s)}[\|(\mathbf{x}_s - \mathbf{x}_i^0) - \epsilon_\theta(\mathbf{x}_i^t, t, \mathbf{z}_s)\|_1]$$

Perplexingly, the first $\mathcal{L}_{\text{cyc}}$ usually seemed to perform slightly better than the second, despite that the first involves an approximation. The reasoning for this is an open question, but our hypothesis is that the model is unable to predict this residual between the input and style images when the input image is obscured by noise. Instead, predicting the residual between the current latent and the style image showed better results.

Gradient Magnitude for Cycle Consistency Given that we're trying to impose pose characteristics from the style image on the input image, explicitly using the edge images is another idea for a loss function. The gradient magnitude image can be used to find edges in an image. Intuitively, minimizing the distance between the gradient magnitude of the style and input images should encourage the diffusion model to generate images that retain pose information from the style image. The gradient magnitude image for an image $\mathbf{x}$ is

$$\|\nabla \mathbf{x}\|_2 = \sqrt{\left(\frac{\delta \mathbf{x}}{\delta x}\right)^2 + \left(\frac{\delta \mathbf{x}}{\delta y}\right)^2}$$

Based on this, we can define a cycle consistency loss function based on the gradient magnitude as

$$\mathcal{L}_{\text{cyc}} = \mathbb{E}_{\mathbf{x}_i^0, t, \mathbf{x}_s, \mathbf{x}_i^t \sim \mathbf{x}_i^0 + \epsilon_{0:t}^{\text{scaled}}, \mathbf{z}_s \sim \mathcal{E}(\mathbf{x}_s)}[\|(\|\nabla \mathbf{x}_s\|_2 - \|\nabla \mathbf{x}_i^0\|_2) - \epsilon_\theta(\mathbf{x}_i^t, t, \mathbf{z}_s)\|_1]$$

This is calculated on the grayscale version of the two images and results in a black-and-white edge image. This potentially allows the model to focus more on the pose difference in the two images and less on the colors and/or textures. Figure 4 shows an example gradient magnitude image.

(a) Original (b) Gradient Magnitude

Figure 4

Partial Noising With the architecture that we have described to this point, there is no way to manually choose an input image $\mathbf{x}_i$ at inference time. The model will simply choose whatever it decodes from a random Gaussian noise tensor. For this reason, we modified the inference to allow us to specify an input image. Instead of sampling a random noise tensor, we select an input image $\mathbf{x}_i$ and partially noise it, stopping the forward process early. In our case, the full schedule consisted of $T = 1000$ diffusion timesteps, and we noised our input images for the first $t_p = 250$ of those timesteps. Subsequently, we sample from the reverse process starting at the same step t_p in the noise schedule. The choice of t_p is important: intuitively, we want it to be low enough to preserve the class characteristics as well as the obvious non-recoverable characteristics of the image. It should also be high enough, though, to destroy some fine detail in the image, which should be "filled in" with the style characteristics.

Music Representation We suffered from extreme memory constraints in our attempts to generate raw music waveforms. This issue was inflated by two primary factors: first, since waveforms are 1-dimensional, our MaxPooling layers would only decrease the size by a factor of 2 instead of 4. In order to achieve the same compression as 2D MaxPooling, we would have to increase the kernel size, but this compression would lose too much local data. As a result, waveforms require a much deeper network to achieve the same degree of compression. Second, our use of attention, an incredibly memory-intensive operation, exacerbated all of the existent issues with waveforms.

A conventional solution to this problem could be to use a Latent Diffusion Model (LDM) [18]. However, our cycle consistency term's functionality, intuition, and interpretibility are contingent on

the existence of semantic information in the ℓ_1 distance between our input and style images. If we compress waveforms down to some unknown representation, then it is significantly less clear what this metric represents. For this reason, it is important that the representation we choose have some interpretable metric to yield some meaning to $\mathcal{L}_{\text{cyc}}$. As mentioned in the background, we use the Log-Mel spectrogram, normalized to be more suitable for learning.

Vocoder In order to decompress our generated Log-Mel Spectrograms into waveforms, we use the DiffWave [11] Vocoder, which is a diffusion model-based generator for waveforms conditioned on Log-Mel Spectrograms. This vocoder was pretrained on the GTZAN dataset on 2 RTX 8000s for about $800,000$ steps, which took over a week. One potential concern with this vocoder was that it may have slightly overfit to its training data. Specifically, since it was trained to map GTZAN spectrograms to GTZAN tracks, and our model slightly modifies GTZAN spectrograms, we feel DiffWave may have slightly biased the output towards the un-modified GTZAN tracks. For this reason, we needed to increase λ_{cyc} quite a bit to achieve some change in the output that would actually propagate through DiffWave. Unfortunately, though, this also resulted in some unwanted artifacts and distortion in the final output as a result of too much regularization.

5 Results

Image Style Transfer The image-to-image translation diffusion model yielded mixed results. With style transfer, we hoped to retain the background, colors, and textures from the input image while transferring the pose characteristics from the style image. Across the samples we generated (Figures 5, 6), the model does an excellent job of maintaining the background information, colors, and textures from the input image.

However, the model appears to mostly ignore pose information from the style image. Instead, it tends to reshape all the input animals into felines, ergo, it performs Cat-lation (cat + translation).

Figure 5: Image-to-Image translation. Left column: input image, Middle column: style image, Right column: output

We feel this can be explained in part by the dataset we used. The AFHQ training set is composed of 5153 cat images, 4739 dog images, and 4738 wild images. Furthermore, many of the wild images are tigers, lions, and leopards, i.e., wild cats! Given a large skew towards cats in the dataset, it is possible that our loss function learned how to "cheat" on the dataset by making everything look like a cat. Additionally, we realized our loss function may not have been properly normalized and the λ_{cyc} parameter for the regularization term may not have been chosen correctly. Additional testing with different hyperparameters may have yielded different results.

Music Style Transfer Our music style transfer results are available here: `https://www.youtube.com/watch?v=kQ7oCG1an10`. We found the model to be successful at transferring style characteristics between tracks, but it usually created a decent degree of distortion for the reasons mentioned earlier. This distortion could potentially be decreased by training DiffWave on a larger dataset to allow for a smaller λ_{cyc} or with more careful tuning of the noise schedule.

Figure 6: Image-to-Image translation. Left column: input image, Middle column: style image, Right column: output

6 Conclusion

Overall, we did not find that the increased generative power of diffusion models was sufficient to eliminate the need for contrastive learning, patching, and other advanced techniques. We did achieve a decent degree of success in the music case, though, making this task, while still rather difficult, within reach.

Limitations Our audio generation models are constrained to produce 20-second mono tracks with a 22.05kHz sample rate. With more memory, a higher sample rate could be used, stereo tracks could be generated, and the generated tracks could be longer. The quality of these tracks could also be improved with the use of a vocoder designed for music, such as in [4], specifically one pretrained on a large dataset. This would allow for more careful tuning and reduce the potential of the vocoder to overfit to the GTZAN dataset as was aforementioned. Generating raw waveforms would also likely result in higher quality, but we lacked the memory to do so.

Finally, our models could likely be improved with a decent amount of hyperparameter tuning, but we suffered from relative time constraints. That is, there were an extremely large number of hyperparameters (especially in the noise schedule), and training the models just once could take upwards of a day. For this reason, although the models are not badly tuned, we feel that further tuning could greatly improve the quality of the outputs.

Next Steps Aside from the obvious next step of tuning the models further and scaling up to higher sample rates and stereo tracks, one interesting direction could be to come up with more regularization terms to capture alternative qualities of music. These terms could transfer other features in the music, such as key, tempo, or melody.

References

[1] Andrea Agostinelli et al. *MusicLM: Generating Music From Text*. 2023. arXiv: 2301.11325 [cs.SD].

[2] Yunjey Choi et al. "StarGAN v2: Diverse Image Synthesis for Multiple Domains". In: *CoRR* abs/1912.01865 (2019). arXiv: 1912.01865. URL: http://arxiv.org/abs/1912.01865.

[3] Timo I. Denk et al. *Brain2Music: Reconstructing Music from Human Brain Activity*. 2023. arXiv: 2307.11078 [q-bio.NC].

[4] Bruno Di Giorgi, Mark Levy, and Richard Sharp. *Mel Spectrogram Inversion with Stable Pitch*. 2022. arXiv: 2208.12782 [cs.SD].

[5] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[6] *GTZAN Dataset for Music Genre Classification*. URL: https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification.

[7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising Diffusion Probabilistic Models". In: *CoRR* abs/2006.11239 (2020). arXiv: 2006.11239. URL: https://arxiv.org/abs/2006.11239.

[8] Qingqing Huang et al. *Noise2Music: Text-conditioned Music Generation with Diffusion Models*. 2023. arXiv: 2302.03917 [cs.SD].

[9] Ihssane Khallassi, My Alaoui, and A. Jilbab. "SEGMENTATION OF CANCER MASSES ON BREAST ULTRASOUND IMAGES USING MODIFIED U-NET". In: *Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska* 13 (Sept. 2023), pp. 11–15. DOI: 10.35784/iapgos.5319.

[10] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. *HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis*. 2020. arXiv: 2010.05646 [cs.SD].

[11] Zhifeng Kong et al. *DiffWave: A Versatile Diffusion Model for Audio Synthesis*. 2021. arXiv: 2009.09761 [eess.AS].

[12] Hanbit Lee, Jinseok Seol, and Sang-goo Lee. "Contrastive Learning for Unsupervised Image-to-Image Translation". In: *CoRR* abs/2105.03117 (2021). arXiv: 2105.03117. URL: https://arxiv.org/abs/2105.03117.

[13] Mark Levy et al. *Controllable Music Production with Diffusion Models and Guidance Gradients*. 2023. arXiv: 2311.00613 [cs.SD].

[14] Ming-Yu Liu, Thomas M. Breuel, and Jan Kautz. "Unsupervised Image-to-Image Translation Networks". In: *CoRR* abs/1703.00848 (2017). arXiv: 1703.00848. URL: http://arxiv.org/abs/1703.00848.

[15] Max Morrison et al. *Chunked Autoregressive GAN for Conditional Waveform Synthesis*. 2022. arXiv: 2110.10139 [eess.AS].

[16] Paul Pedersen. "The Mel Scale". In: *Journal of Music Theory* 9.2 (1965), pp. 295–308. ISSN: 00222909. URL: http://www.jstor.org/stable/843164 (visited on 04/14/2024).

[17] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. "Searching for Activation Functions". In: *CoRR* abs/1710.05941 (2017). arXiv: 1710.05941. URL: http://arxiv.org/abs/1710.05941.

[18] Robin Rombach et al. "High-Resolution Image Synthesis with Latent Diffusion Models". In: *CoRR* abs/2112.10752 (2021). arXiv: 2112.10752. URL: https://arxiv.org/abs/2112.10752.

[19] Nataniel Ruiz et al. "DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2023, pp. 22500–22510.

[20] Chitwan Saharia et al. "Palette: Image-to-Image Diffusion Models". In: *ACM SIGGRAPH 2022 Conference Proceedings*. SIGGRAPH '22. Vancouver, BC, Canada: Association for Computing Machinery, 2022. ISBN: 9781450393379. DOI: 10.1145/3528233.3530757. URL: https://doi.org/10.1145/3528233.3530757.

[21] Junhao Wang et al. “An Unsupervised Methodology for Musical Style Translation”. In: *2019 15th International Conference on Computational Intelligence and Security (CIS)*. 2019, pp. 216–220. DOI: 10.1109/CIS.2019.00053.

[22] Tengfei Wang et al. *Pretraining is All You Need for Image-to-Image Translation*. 2022. DOI: 10.48550/ARXIV.2205.12952. URL: https://arxiv.org/abs/2205.12952.

[23] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. *Adding Conditional Control to Text-to-Image Diffusion Models*. 2023. DOI: 10.48550/ARXIV.2302.05543. URL: https://arxiv.org/abs/2302.05543.