Audio Style Transfer

Roger Finnerty Brennan Mahoney Amruth Niranjan Sanford Edelist Zane Mroue
{jrfinn, bmm1, amruth, edelist, zanem}@bu.edu

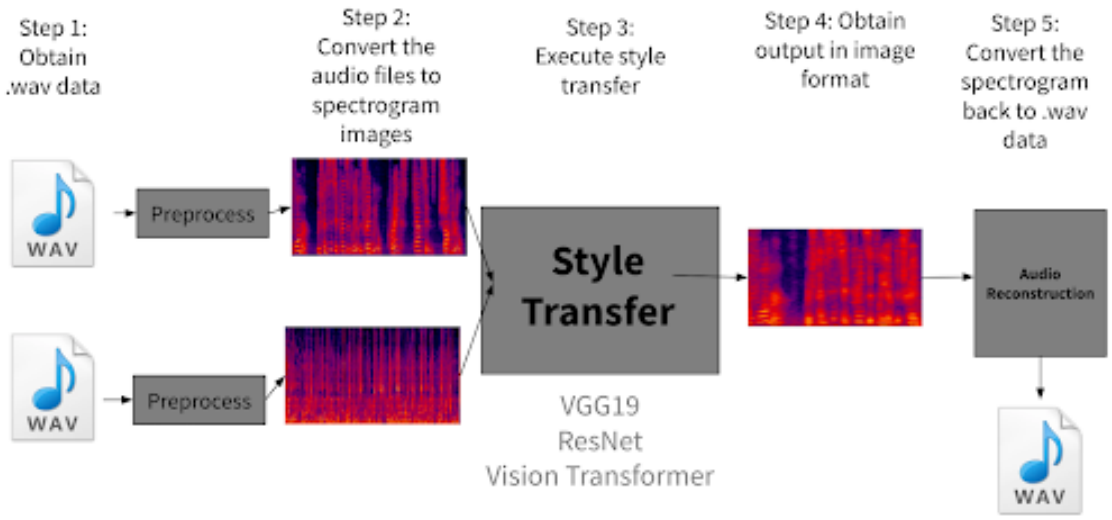

Figure 1: Audio Style Transfer Pipeline

1 Task

With the rise in popularity of generative music tools and new uses for generated audio at the time this paper was written, the initial motivation for this paper is to change the gender and accent of song acapellas. However, this end goal poses significant challenges to the process, including computationally expensive training procedures, difficulty capturing nuances across accents, and high fidelity audio reconstruction from Mel spectrograms.

Given these challenges, we revised our task as follows: execute style transfer on short 5-to-10-second audio samples (with styles for gender, instrument, accent transfer), compare 4 models for style transfer, implement a deep-learned model and classical method for audio reconstruction from spectrogram, and perform a thorough qualitative evaluation of the results.

2 Related Work

The work was initially inspired by the neural style transfer algorithm proposed by Gatys et al. in [2]. This method generates an image with the goal of preserving the content of one image while applying the style of another image; one popular application is the application of textures from famous paintings to photographs, as show in Figure 2.

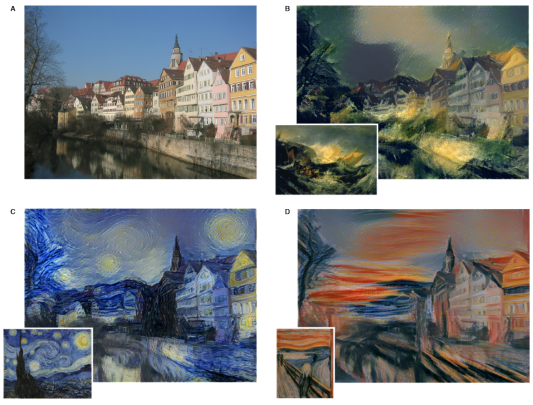

Figure 2: Style Transfer Examples [2]

This method generates an image by optimizing a loss composed of content and style loss terms, corresponding to the differences between the generated image and the content and style images. Style and content representations are captured by the feature maps at intermediate layers of the popular image classification network VGG-

19, which is built from convolutional and pooling layers. The content loss at some layer l in the network is then the squared distance between the VGG feature representations of the generated and content images at that layer. Style representations are computed using a Gram matrix, which is the inner product between the vectorised feature map at some layer. The style loss is then the squared distance between the style representations of the generated and style images. Starting with a pure white noise image, this algorithm performs gradient descent to iteratively refine the image to take on the desired style and content.

We imagined that this approach could work to do style transfer on spectrograms created from content and style audio clips to do audio style transfer. Research on work in audio style transfer ([3] and [5]) validated our idea that style transfer via spectrograms was indeed a feasible and popular approach. Thus we set out conduct our own experiments in this field, beginning first by using method from [2] to do style transfer on Mel spectrograms. In order to reconstruct audio from Mel spectrograms, we benefited greatly from the open-sourced work published by Gaurav K. Verma on Kaggle on this topic [6]. This architecture used WaveNet [4], a deeply learned model for generating raw audio published by Google DeepMind in 2016. The model is fully probabilistic and autoregressive, generating each new audio sample based on the joint probability of all previous samples. In the Kaggle implementation, the WaveNet model was trained on the LJ Speech dataset (more in Section 4) and conditioned to generate audio based on spectrograms - exactly what we intended to do (see Equation 1, where h is the spectrogram).

$$p(x|h) = \prod_{t=1}^{T} p(x_t|x1, ..., x_{t-1}, h) \tag{1}$$

Furthermore, we were able to download weights of the model pretrained on the speech dataset and used the same architecture, which provided a starting point for our initial experiments in audio style transfer. We strived to compare other methods for style transfer, choosing a simple 1-layer CNN as well as using ResNet50 and Vision Transformer backbones instead of the VGG in the method from [2]. We also wanted to compare the reconstructed audio from WaveNet with audio reconstruction via the classical Griffin-Lim algorithm as a baseline.

3 Approach

3.1 Pipeline Overview

As illustrated in Figure 1, our approach receives content and style in '.wav' format, preprocesses the audio, creates spectrograms, performs style transfer on the spectrograms, and outputs a generated '.wav' file.

We use the .wav file format due to its uncompressed and lossless qualities so that the spectrograms obtained do not represent compressed audio. The preprocessing refers to resampling the audio to 16kHz. Note that because the content and style audio files do not have the be the same length; while the style transfer network requires content and style images to be the same size, we wrote a function that pads whichever spectrogram was smaller (in the time dimension) by pasting copies of itself to match in this dimension.

Converting the audio into spectrograms is done by applying a Short-Time Fourier Transform (STFT). This provides a complex-valued matrix where each row corresponds to a frequency bin and each column corresponds to a time segment, such that each position *(i, j)* represents the amplitude and phase information for frequency bin *i* at time segment *j*. We then extract the phase information and convert the magnitudes of the resulting matrix in to a logarithmic scale to reduce the dynamic range in magnitudes.

The execution of the style transfer follows, using one of four models (later discussed) that generates a new spectrogram containing the style transfer results. Lastly, we reconstruct the audio from the generated spectrogram using our methods to be discussed. The result is now ready for evaluation.

3.2 Baseline Model

As our initial approach for the style transfer, we proposed a simple single-layer 2D convolutional neural network (CNN). The model consists of a 3x1 kernel filter in the 2D convolutional layer, followed by a leaky ReLU with a slope coefficient of 0.2. We randomize the weights and fix the biases. This method of reconstruction utilizes the traditional Griffin-Lim algorithm, as opposed to a deeply learned approach. We define the following loss functions

$$J_{\text{content}} = \frac{1}{4 \times m \times n_C \times n_H \times n_W} \sum (\mathbf{a_C} - \mathbf{a_G})^2 \tag{2}$$

$$J_{\text{style\_layer}} = \frac{1}{4 \times (n_C^2) \times (n_H \times n_W)} \sum (\mathbf{G_S} - \mathbf{G_G})^2 \tag{3}$$

where a_C is the feature representation of the content image, a_G is the feature representation of the generated image, m is the batch size, n_C is the number of channels, n_H is the height of the tensor, n_W is the width of the tensor, G_S is the Gram matrix of the style representation of the target image, and G_G is the Gram matrix of the style representation of the generated image. We train the model over 20000 epochs with a learning rate of 0.002, style weight of 1, and content weight of 100.

Based on auditory observation, the results (found in the Git repository) indicate that our simple baseline model was effective.

Figure 3 shows the loss curve for a male voice to instrument transfer, providing further indication that our model succeeds to perform our desired task. Given

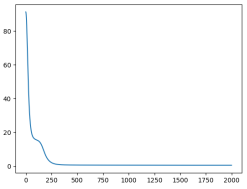

Figure 3: Loss curve of initial baseline model for voice to instrument transfer

the positive outcome from this model, we continue to propose "deeper" models to achieve better results.

3.3 VGG19

The second method that we used for the style transfer was a VGG19 CNN [2]. The architecture for this method is shown below:

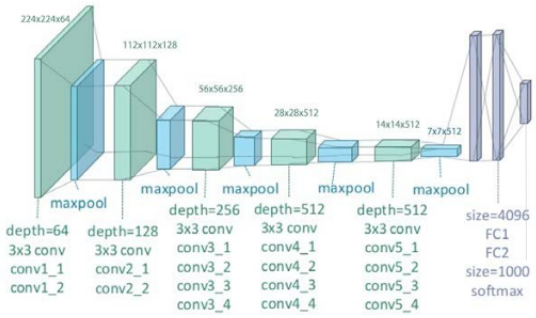

Figure 4: VGG19 Architecture

VGG19 is a deep image classification model that has the ability to differentiate and manipulate the content and style representations at different layers of the network. Model weights for VGG pretrained on the ImageNet dataset were imported from the torchvision.models library. The style transfer process uses loss functions to guide the optimization: a content loss ensures the output matches the content of the target image, while a style loss makes sure the artistic style of the reference image is captured. By minimizing these loss functions, the model iteratively adjusts the target image until the desired stylistic transformation is achieved. The loss functions are shown below:

$$\mathcal{L}_{content}(p, x, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2 \tag{4}$$

$$\mathcal{L}_{style}(a, x) = \sum_{l=0}^{L} w_l E_l \tag{5}$$

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2 \tag{6}$$

$$\mathcal{L}_{\text{total}} = \alpha \mathcal{L}_{\text{content}} + \beta \mathcal{L}_{\text{style}} \tag{7}$$

In (4), F_{ij}^l and P_{ij}^l are the feature representations of the generated and original content images at layer l. The style loss in 5 is computed as a weighted sum of the normalized squared differences between the style representations of generated (G_{ij}^l) and original style (A_{ij}^l) images. α and β are weighting factors that control the relative importance of content and style in the synthesized image, respectively. These parameters can be tuned to prioritize either fidelity to the content or adherence to the artistic style.

3.4 ResNet

The third method that was utilized for this project was neural style transfer with a ResNet50 backbone. The architecture for this method is shown in Figure 5:

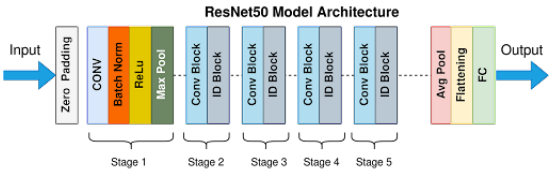

Figure 5: ResNet50 Architecture

We adapted a ResNet50, a deep convolutional neural network known for its residual blocks that help in avoiding the vanishing gradient problem, for style transfer. Like VGG19, ResNet50 can be used to separate and recombine content and style from different images. The pretrained ResNet50 would extract deep features from both the content image and the style reference. Loss functions specific to content and style are applied to guide the optimization process, adjusting the target image iteratively to blend the content of one image with the style of another, effectively leveraging ResNet50's powerful feature extraction capabilities. The loss functions for are shown below:

$$\mathcal{L}_{\text{content}} = \frac{1}{H \times W \times C} \sum_{i,j,k} \left(F_{\text{content}}^{ijk} - F_{\text{target}}^{ijk} \right)^2 \tag{8}$$

where F_{content} and F_{target} are the activations of the predetermined layer for the content and target images, respectively, and H, W, C denote the dimensions of the respective feature maps.

$$\mathcal{L}_{\text{style}}^l = \frac{1}{4C_l^2 H_l^2} \sum_{i,j} \left(G_{\text{style}}^{l,ij} - G_{\text{target}}^{l,ij} \right)^2 \tag{9}$$

where G_{style}^l and G_{target}^l are the Gram matrices of the activations for the style and target images at layer l, respectively. C_l and H_l are the number of channels and the size of the feature map at layer l, and the Gram matrix G is defined as $G_{ij} = \sum_k F_{ik} F_{jk}$, with F representing the feature map matrix. Equation (7) was used to compute the total loss.

3.5 Vision Transformer

The fourth and final model we used as an approach to achieve our task was a vision transformer model. The architecture for this model is shown in Figure 6.

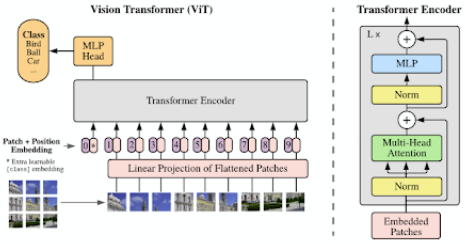

Figure 6: Vision Transformer Architecture

The Vision Transformer (ViT), originally modeled in [1] can be used for style transfer because it is designed to process images as learnable sequences of patches. This unique attribute theoretically allows ViT to understand both local and global image features as effectively as baseline models for similar tasks. As defined in the original paper, the blocks transform the image $x \in \mathbf{R}^{H \times W \times C}$ (where *H* and *W* define the size of the image, and *C* represents the number of channels) into a sequence of patch embeddings $x_p \in \mathbf{R}^{N \times (P^2 \cdot C)}$, where now (*P, P*) represents the resolution of each image patch and $N = (H \times W)/P^2$. For style transfer, the content and style loss functions we used include

$$\mathcal{L}_{\text{content}}(x_p) = \frac{1}{NP^2C} \sum_{i,j} \left(Q^{ij}_{\text{content}} - Q^{ij}_{\text{target}} \right)^2 \quad (10)$$

where Q_{content} is the content feature embedding for the image patch p and Q_{target} is the latent representation of the target image,

$$\mathcal{L}_{\text{style}}(x_p) = \frac{1}{NP^2C} \sum_{i,j} \left(G^{ij}_{\text{style}} - G^{ij}_{\text{target}} \right)^2 \quad (11)$$

where G_{style} and G_{target} are the Gram matrix outputs for the style feature embedding of image patch p and latent representation of the target image respectively, with the Gram matrix being defined earlier, and a slightly modified version of (7)

$$\mathcal{L}_{\text{total}}(x_p) = \alpha L_{\text{content}}(x_p) + \beta L_{\text{style}}(x_p) \quad (12)$$

where $\mathcal{L}_{\text{style}}(x_p)$ represents the total loss for patch p. The result for the most optimal loss follows using a combination of selected patches that minimize the loss the most.

The means by which content and style features were obtained differed from a standard convolutional approach. The ViT we used lacks a "layered" approach akin to the notion of convolutional models, and instead features twelve (nine in image) patch and position embedding blocks that are fed into a transformer encoder for a more optimized classification. Because of this, our approach opted to extract content and style features from multiple combinations of the block patches for content and style features, eventually settling with the combination of blocks that yielded the lowest loss.

Ideally, these functions optimize the output image to reflect the content of one image and the style of another. However, the output of the style transfer performed by our method for the ViT yielded negative results: garbled, noisy spectrograms which yielded similarly poor audio. In the future, we may revisit the ViT with a different approach where content and style feature extraction occurs inside the transformer encoder itself (perhaps as the output to the multi-headed self attention module), as opposed to exclusively using the outputs of the patch-embedding blocks.

3.6 WaveNet Audio Reconstruction

For the three deeply learned style transfer models, we used the WaveNet to convert the output spectrogram back to audio. The architecture for this is shown in Figure 7.

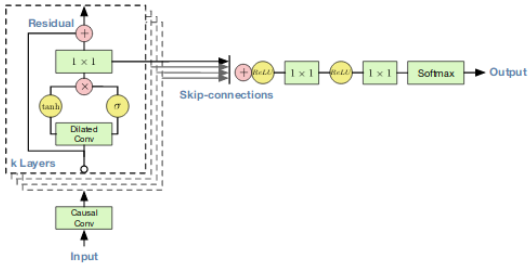

Figure 7: Unconditional Wavenet Architecture

We adapted a WaveNet model to reconstruct audio from a spectrogram, which uses a deep convolutional neural network architecture that utilizes convolutions to capture the temporal context of audio signals. For the local spectrogram-to-audio reconstruction task, the WaveNet takes the spectrogram as input, interpreting it as a compressed representation of audio features. The model then predicts audio waveforms sample-by-sample, transforming the frequency-time information of the spectrogram back into an audio signal.

3.7 Super Resolution

In an attempt to improve the generated audio, we passed the style and content clips into a deeply learned super-resolution model, which improves the resolution of the images. An example output is included in the following figures.

The super resolution was done using the Enhanced Super Resolution GAN from [7], which has an open-sourced notebook published. This ultimately did not produce any noticeably improved results because the pretrained weights that were imported for the WaveNet

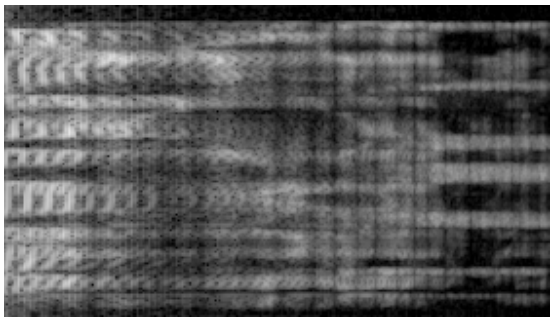

Figure 8: Content Pre-Super Resolution

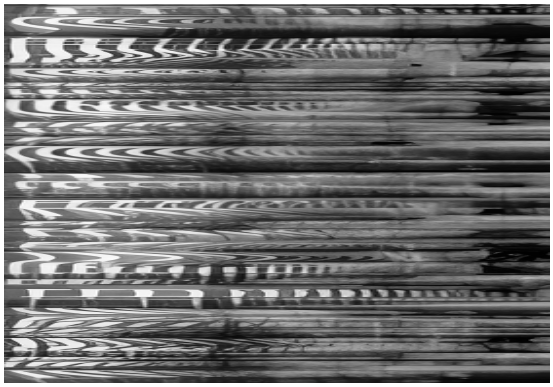

Figure 9: Content Post-Super Resolution

model expected a certain value for the horizontal (frequency bin) dimension of the spectrogram, so super resolution would likely required training the model on these spectrograms. It is also probable that the super resolution does not preserve the true structure of the audio that is contained in the spectrograms produced by the STFT.

4 Datasets

The VGG19 and ViT models were both pretrained on the ImageNet database, which features over 14 million images of general objects, and is regarded as an all-encompassing benchmark dataset for most computer vision tasks. For the content audio, we used the LJ Speech Dataset, a public domain speech dataset taken from audiobooks. This dataset features over 13,000 short audio clips ranging from 1 to 10 seconds. The style audio, on the other hand, consisted of a variety of unique audio clips, ranging from unique accents, gendered pitches, and even instruments. These audio clips were less constrained allowing for a greater flexibility in the types of audio manipulations we could explore. By leveraging these datasets we could effectively train and validate our models, ensuring they are capable of style transfer that change the inherent auditory qualities. The next steps in our project include refining the models to better handle the subtleties of audio data, particularly how they handle unique characteristics of spoken language such as different accents. We also plan to expand our datasets to include more nuanced variations of speech and musical styles to further enhance the versatility of our style transfer processes.

5 Evaluation Metrics

In the WaveNet paper, the authors use a Mean Object Score (MOS) to evaluate the performance of their audio generation. This metric asks subjects to rate the "naturalness" of the generated audio on a five-point Likert scale (1:Bad, 2:Poor, 3:Fair, 4:Good, 5:Excellent). This affirms our notion that there is no standard means of evaluating generative audio models. In order to evaluate our results, we employed a similar method by surveying numerous randomly selected Boston University students and asking them to give a rating on a scale of 1-10 of each of our generated audio clips. The results are included in Figures (13), (14), and (15) in the following section.

6 Results

For the VGG19 model we plotted the loss curves for the style and content loss as shown in the figures below:

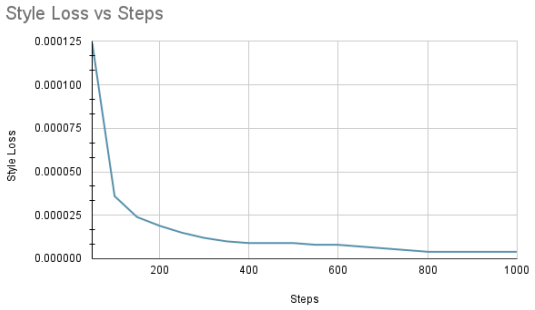

Figure 10: Style loss curve for the VGG19 model when doing an accent audio style transfer

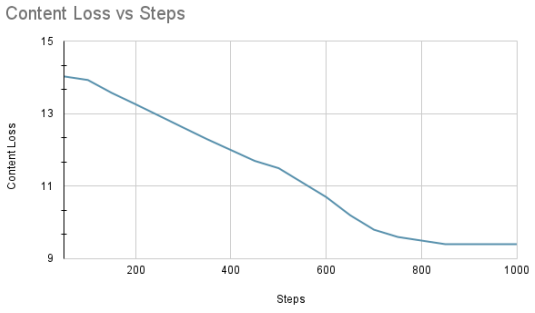

Figure 11: Content loss curve for the VGG19 model when doing an accent audio style transfer

Furthermore, as referenced earlier, our principle means of evaluation was through a Google Form survey of 57 randomly sampled Boston University students using a MOS evaluated from a score of 1 (incredibly poor) to 10 (flawless). The figures below showcase the

distribution of the public's responses for each model that yielded audible results (all except for the ViT).

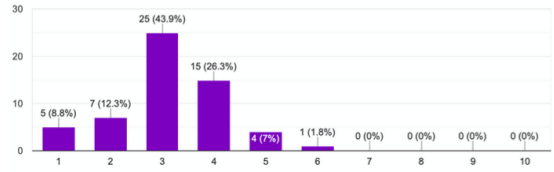

Figure 12: Distribution of responses for the model using the 1-layer CNN + STFT and the Griffin-Lim algorithm for reconstruction

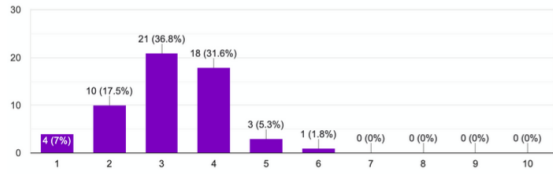

Figure 13: Distribution of responses for the model using the VGG19 backbone and WaveNet reconstruction

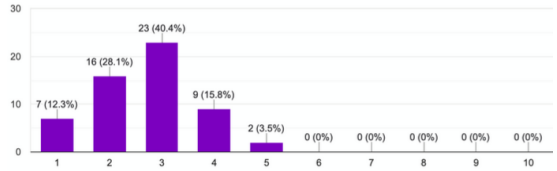

Figure 14: Distribution of responses for the model using the ResNet backbone and WaveNet reconstruction

Our audio results are displayed for view in our GitHub under the "Results" folder. There are results for the 1-layer CNN, VGG19, and the ResNet. For each of these models, we tried 3 different audio-style transfers: gender, instrument, and accent.

7 Conclusion

In conclusion, neural style transfer with spectrograms using our approach can be used to introduce new styles of audio to pre-existing content audio at a high-level, however the spectrograms alone fail to capture the lower-level, subtler differences between accents (for speakers of the same gender). Although the overall effectiveness of our approach did not yield incredibly accurate or overwhelmingly positive results, the results of our experiments still offer valuable insight into the future of this problem. We believe that the accuracy of our results would improve significantly if, in the future, training was targeted to one specific style transfer task (accents, timbre, pitch, gender, etc). Some future ideas that we can work on for this project are retraining WaveNet with super resolution spectrograms to potentially better capture audio information contained in the spectrogram. Additionally, we look forward to developing new, more optimized approaches to perform NST using recurrent architectures like ViT. Finally, attempting supervised learning by add a label to audio clip (ground truth accent), as demonstrated by conditional diffusion.

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[2] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

[3] Eric Grinstein, Ngoc QK Duong, Alexey Ozerov, and Patrick Pérez. Audio style transfer. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 586–590. IEEE, 2018.

[4] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[5] Marco Pasini. Melgan-vc: Voice conversion and audio style transfer on arbitrarily long samples using spectrograms. *arXiv preprint arXiv:1910.03713*, 2019.

[6] Gaurav K. Verma. How to convert audio to mel-spectrogram to audio, 2021.

[7] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European conference on computer vision (ECCV) workshops*, pages 0–0, 2018.

A Detailed Roles

See the breakdown of our project by individual contribution in Table 1.

Name	Task	File Names	No. Lines of Code
Roger	VGG + Wavenet, super resolution	vgg-wavenet.py, style-transfer.py, wavegen.py, utils.py, audio-utils.py	452
Brennan	VGG + Wavenet, Spectrogram to Audio	vgg-wavenet.py, style-transfer.py, wavegen.py, utils.py, audio-utils.py, spectrogram-to-audio-brennan	753
Amruth	WaveNet + ViT	vgg-wavenet.py, Vision-Transformer.ipynb	450
Jack	CNN + Griffin-Lim	CNN-style-transfer.ipynb	297
Zane	ResNet + Wavenet	EC523-FinalProject-Script-ResNet.ipynb	436

Table 1: Summary of Tasks

B GitHub Repository

See our full project repository at: https://github.com/rogerfinnerty/Audio-Style-Transfer/tree/main