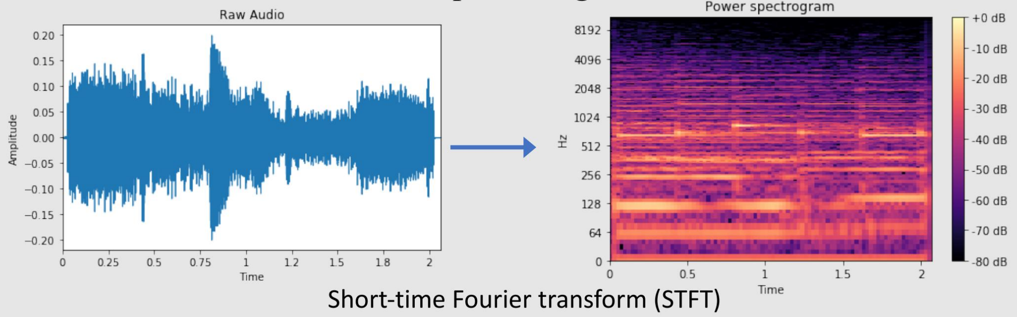
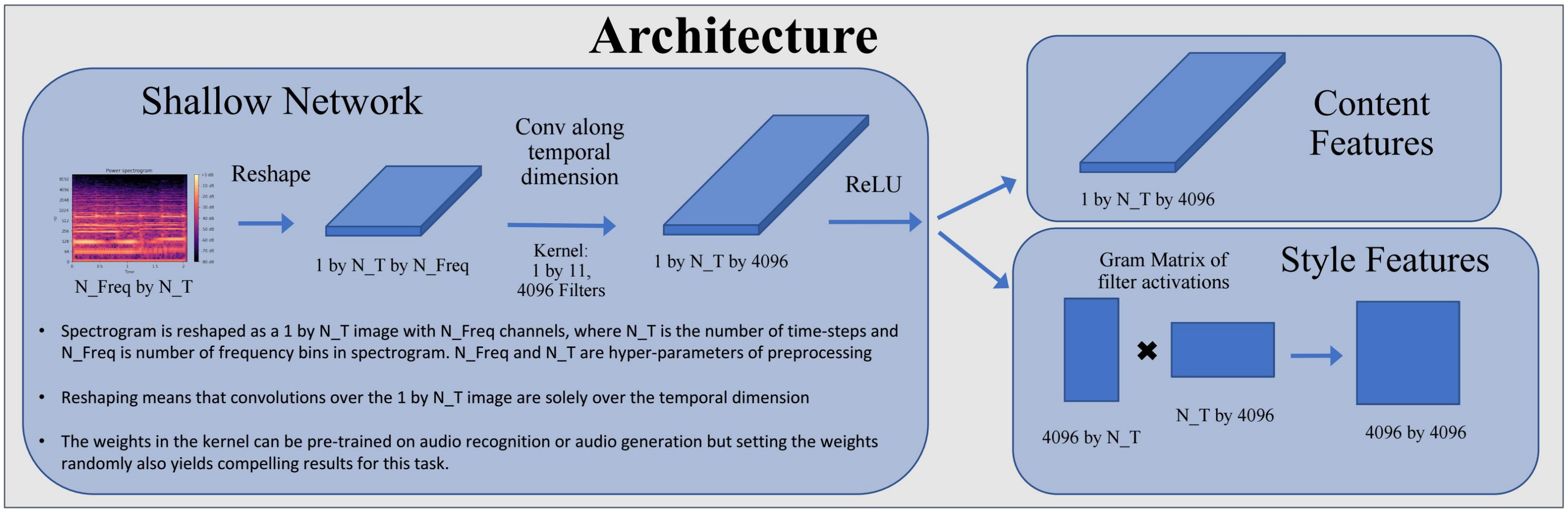
Pre-processing

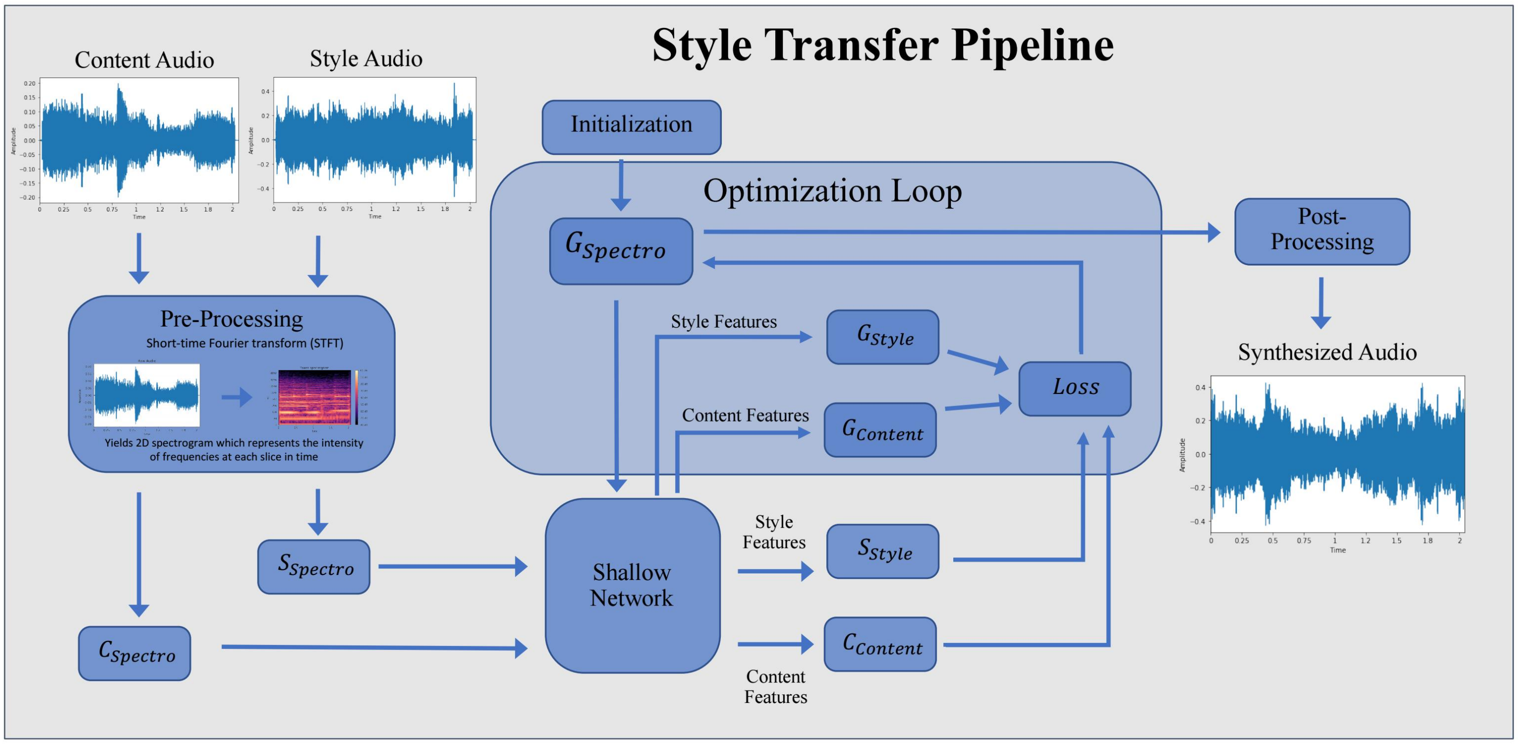


Short-time Fourier transform (STFT)

- Input raw audio (as .mp3) transformed to 2D spectrogram via STFT, which represents the intensity of frequencies at each slice in time
- All audio is clipped to exactly 2 seconds for a constant spectrogram shape



- Spectrogram is reshaped as a 1 by N_T image with N_Freq channels, where N_T is the number of time-steps and N_Freq is number of frequency bins in spectrogram. N_Freq and N_T are hyper-parameters of preprocessing
- Reshaping means that convolutions over the 1 by N_T image are solely over the temporal dimension
- The weights in the kernel can be pre-trained on audio recognition or audio generation but setting the weights randomly also yields compelling results for this task.


Style Transfer Pipeline
Content Audio
Style Audio
Initialization
Optimization Loop
$G_{Spectro}$
Post-Processing
Pre-Processing
Short-time Fourier transform (STFT)
Yields 2D spectrogram which represents the intensity of frequencies at each slice in time
Style Features
G_{Style}
$Loss$
Content Features
$G_{Content}$
Synthesized Audio
$S_{Spectro}$
Shallow Network
Style Features
S_{Style}
$C_{Spectro}$
$C_{Content}$
Content Features
Amplitude
Time

Stanford University
CS 230

Style Transfer on Instrumental Music

Austin Narcomey

Motivation & Problem

- Neural Style Transfer [2] has been highly successful on 2D images, but applying to 1D audio signals with overlapping sounds and long-term dependencies has proven challenging. This projects explores models of audio style transfer and tests assumptions in existing literature

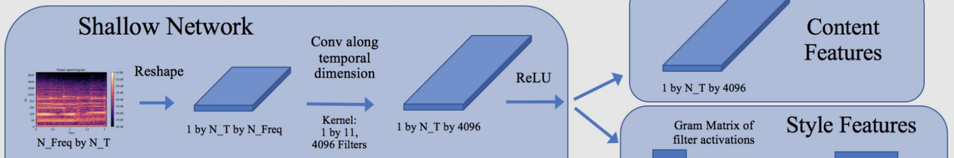
Parameters and Results

- Number of Filters: This model, developed by Ulyanov and Lebedev [1], uses an unusually high number of filters, but scaling the number of filters down to 1024 caused the generative network to be entirely unable to capture style features
- Initialization of Output: Gatys et al. [2] used random initialization of output so that the generated output be forced to deviate from the original content, but in this application where the images are spectrograms the output easily acquired features of the style recording with content initialization, which saved runtime. Grinstein et al. [3] reported better results with content initialization and no content loss, but in tests with this architecture the content was lost completely in the absence of content loss.
- Content and Style Weight: This model was prone to taking on too much structure of the style recording and losing the desired content, so a 1000:1 ratio of content to style weight worked well to correct. This is among the most sensitive hyper-parameters and settings on one combination of style and content can yield poor performance on other combinations
- Pre-trained Weights: Using pre-trained weights offered no observable advantage; however, this shallow network architecture trained extremely slowly on an auxiliary instrument classification task and had poor performance on that task. Thus it is unclear whether more extensive training would improve performance or random weights yield equivalent performance to pre-trained weights

References

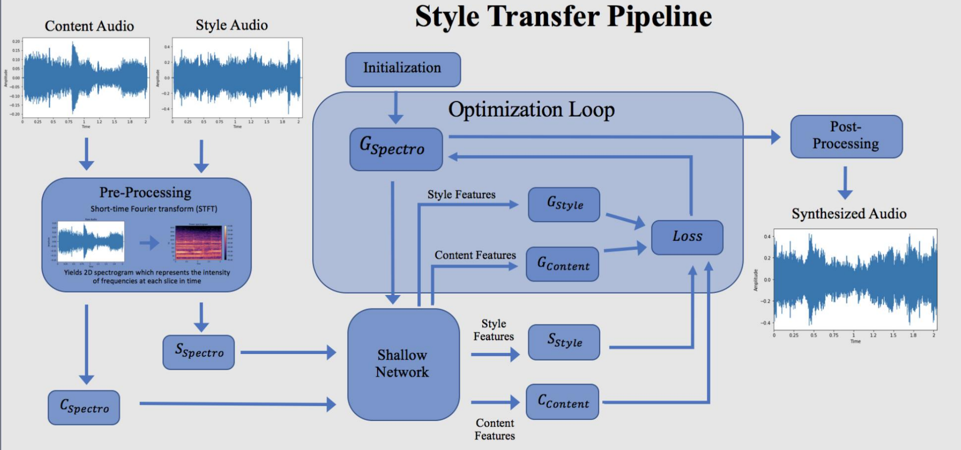
- [1] Dmitry Ulyanov and Vadim Lebedev, “Audio texture synthesis and style transfer,” 2016
- [2] Leon A Gatys, Alexander S Ecker, and Matthias Bethge, “Image style transfer using convolutional neural net-works”, 2016
- [3] Eric Grinstein, Ngoc Q. K. Duong, Alexey Ozerov and Patrick Perez, "Audio Style Transfer", 2017

Architecture



- Spectrogram is reshaped as a 1 by N_T image with N_Freq channels, where N_T is the number of time-steps and N_Freq is number of frequency bins in spectrogram. N_Freq and N_T are hyper-parameters of preprocessing
- Reshaping means that convolutions over the 1 by N_T image are solely over the temporal dimension
- The weights in the kernel can be pre-trained on audio recognition or audio generation but setting the weights randomly also yields compelling results for this task.

Style Transfer Pipeline



Stanford University
CS 230

Style Transfer on Instrumental Music

Austin Narcomey

Motivation & Problem

●

Models & Pipeline

● **Bayesian Network:** Consists of variables that a film-maker would choose (such as budget, genre, etc.) and dependent variables revenue and budget. Users query the network to learn how film variables affect each other.

● **Regression:** After experimenting with variables in the Bayesian Network, a studio can request a prediction of their film's revenue and rating, which is computed via regression on qualities known pre-release. This provides an evaluation of their film to guide development.

Moving Forward

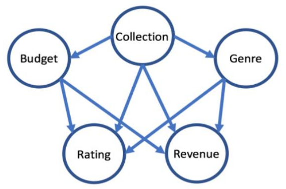
● We plan to experiment with other models like neural network, improve feature extraction, and implement the user interaction pipeline.

Approaches

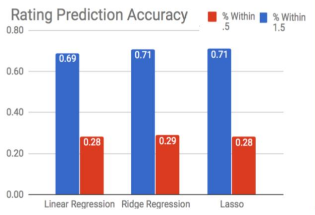
● **Processing Data:** Dataset of ~45,000 films: for text data, bag of words features were used, and for numeric data all features were normalized to handle varying scales. Experimented with methods of handling nonlinear features such as higher-order terms

● **Learning Predictor:** Variety of learning algorithms such as Linear Regression, Ridge Regression, and Lasso in order to regularize the weight vector to prevent overfitting. 10-Fold Cross-validation was used to evaluate changes to the model.

● **Building Bayesian Network:** Designed from domain knowledge about films, local distributions learned with MLE.



Example Query: What is the probability that a film achieves a rating of at least 7 given that it is an action movie with a budget under $5 million?



Challenges

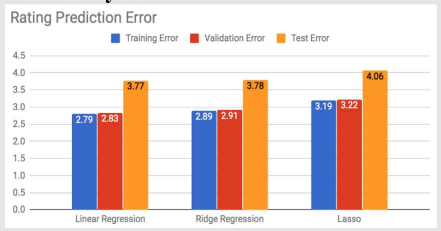
● **Nonlinearity of features:** Features such as budget have nonlinear relationships (such as saturated and non-monotonic) with outputs.

● **Missing Data:** Some fields are not populated in the dataset.

● **Feature Extraction from Text:** Many methods to choose from to extract numerical features from text-based data.

● **Bayesian Network Design:** Building network to capture dependencies between variables.

Results & Analysis



● Using linear regression, ridge regression, and the lasso, the lowest test error (using root-mean-square error) was achieved with linear regression.

● Since lasso shrinks the coefficients of variable weights to 0, this shows most of the predictors used in prediction were useful in predicting movie rating. To contrast, shrinking the coefficients partially (not fully to 0) was more effective.

● Ridge regression gave the highest accuracy on rating within +/- 1.5 and +/- .5 stars.

● As an example, we overestimated a conventional high-budget blockbuster, War for the Planet of the Apes, with true rating 7.0 (out of 10) and revenue $370M, by predicting a maximum rating 10 (out of 10) and revenue $735M.