CS 753: ASR Project

170050064, 170050007, 170070015

1 Problem Statement

The goal is to create a music genre transfer model using an Encoder-Decoder model, and this involves the following steps :-
1. Preprocessing the GT-ZAN dataset to convert audio files to mel-spectrograms.
2. Train the Encoder-Decoder model (with an embedded discriminator) using the spectrograms.

2 Metrics

Evaluation of generative models is usually done by either perceptually or by training a separate classifier on the original data and seeing how well the generator confounds this classifier. We use only the former approach in the present case.

3 Data Exploration

1. GT-ZAN dataset
The dataset consists of 30 second song samples of different genres. We consider two genres for the task at hand here - Classical and Jazz. We convert these to mel-spectrogram representations using the 'librosa' library. The converted spectrograms are converted back to audio using the Griffin-Lim algorithm.

4 Data Preprocessing

Since we are trying a cnn implementaion we need to transform audio features to 2D space, now for this purpose we use the mel-spectogram although we could have used midi features as well. Now for extraction of mel soectogram we use the Python library librosa. we see that by default it extracts feature with 128 mel nbins. Also note that since we have a 30 second audio and use a sampling rate of 22 khz, for each file we get $128/cross1293$ dimensional mel spectogram. since we prefer passing a square image we extract the mel spectogram with mel bins equal to 256 and then pass 256*256 dimensional image to the network, which correspond to roughly 6 seconds of audio.

5 Implementation

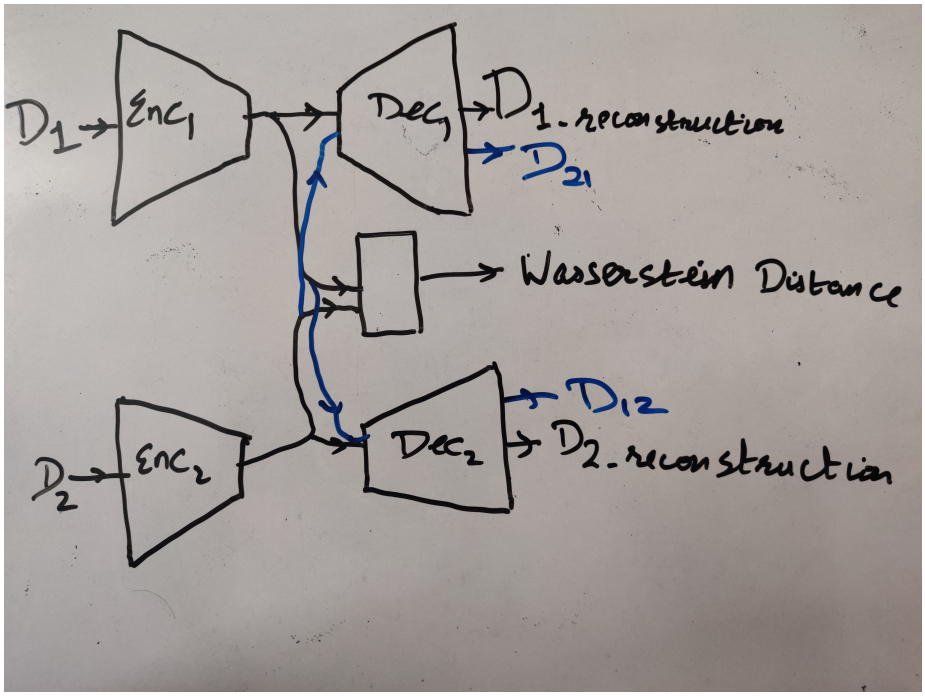


Figure 1: Model Architecture

The implementation process can be split into 3 main stages:-

1. Encoding stage
2. Dicriminating stage
3. Decoding stage

5.1 Stage-1 Encoding

We start with 2 different domains one for each style. Now we use are two encoders for each domain, the task of these encoders is to learn style independent latent space information which effectively fools the discriminator. We do this to ensure that the latent spaces of each genre are aligned.

5.1.1 First attempt- U-Net architecture

We initially utilised a U-Net based backbone to act as the encoder-decoder architecture. The model is a high capacity model used for image-segmentation, and utilizes skip connections between the encoder and decoder. This leads to

a better reconstruction, however, the assumption of shared latent space breaks down due to this.

5.1.2 SegNet

Now since we do a pixel-wise reconstruction, we use a SegNet backbone and our encoders are the encoder part of that backbone. SegNet is shown to perform good on pixel-wise segmentation tasks in the vision community. Further, it suits our task since it does not make use of skip connection that transfer data from encoder to decoder. Instead SegNet only transfers the Max-Pool indices of the down-sampling layers across the encoder and decoder, apart from the latent space representations.

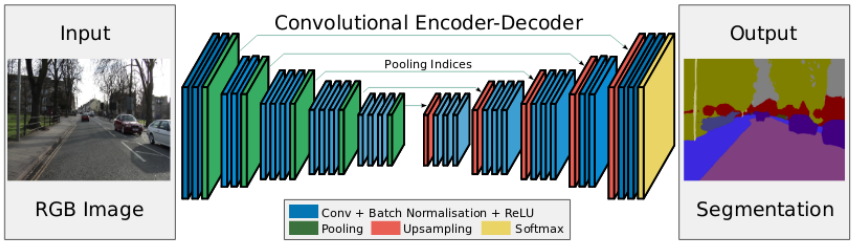


Figure 2: SegNet Architecture

5.2 Stage-2 Discriminator

Discriminator is a simple fully connected network with wasserstien loss. Wasserstien loss is shown to be superior to binary cross-entropy based loss and it provides stable gradients for effective alignment.

5.3 Stage-3 Decoding

Decoder contains the SegNet decoder part and it does pixelwise reconstruction conditioned on the latent space representations. These decoders are supposed to model the style of an audio conditioned on the latent space representation to recover the original audio.
A TensorFlow implementation of our code can be found at this link

6 Training procedure and losses

The entire system is trained like two independent autoencoders, with an additional objective of aligning the latent spaces. We use $\mathcal{X}$ for the original image and $\hat{\mathcal{X}}$ for it's reconstruction. In summary,

$$\mathcal{L}_{enc} = \mathcal{L}_{Recon} - \mathcal{L}_{disc}$$
$$\mathcal{L}_{dec} = \mathcal{L}_{Recon}$$
$$\mathcal{L}_{disc} = \mathcal{L}_{disc}$$

$$\mathcal{L}_{Recon} = \frac{\Sigma_{b=1}^{B}\Sigma_{i,j}|\mathcal{X} - \hat{\mathcal{X}}|}{B}$$

The optimization is done by minibatch gradient descent using an Adam optimizer. We train both the autoencoders as well as the discriminator together, having a gradient reversal layer to propagate the discriminator loss back to the encoders.

6.1 Discriminator loss

We used the Wasserstein distance, using the Kantorovic dual form of the loss to train the discriminator rather than cross entropy. Here $\mathcal{C}_s$ and $\mathcal{C}_t$ represent the source and target encoders and $\mathcal{D}$ represents the discriminator(Critic for wasserstien case) which is a K-Lipschitz function.

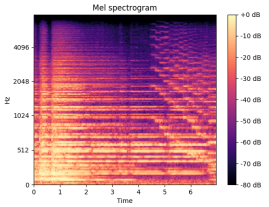
$$\mathcal{L}_{disc} = \sup_{||\mathcal{D}||_{L_P} \leq K} \frac{1}{B}\sum_{b=1}^{B} D(\mathcal{C}_s(x_s)) - \frac{1}{B}\sum_{b=1}^{B} D(\mathcal{C}_t(x_t)) \tag{1}$$

6.2 Reconstruction loss

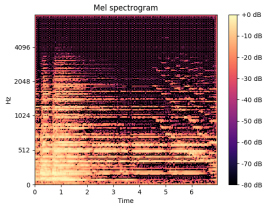
We experimented with both L_1 and L_2 norm. Using L_1 norm for pixel-wise reconstruction gave better results, i.e. more stable training and convergence since the network had gradients to work with near 0.

7 Results and discussion

The U-Net performs almost perfect reconstruction, while SegNet suffers on that front. However, U-Net learns the identity function for both the auto-encoders, essentially turning the genre transfer into an identity mapping.
A failure mode of SegNet happens because the network is of low-capacity, and training with reconstruction loss means that a lot of finer details, i.e. frequencies having low values in the spectrogram are zeroed out, leading to very sparse outputs. This problem needs to be resolved in order to get better results. We also find that the generated outputs have certain traits of the input, roughly having the same "tune", but they have a "tinny" sounding output due to missing frequencies. Some results can be found at this link.

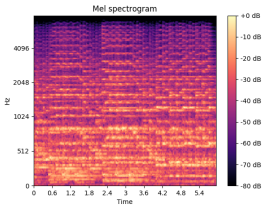


(a) Original Spectrogram

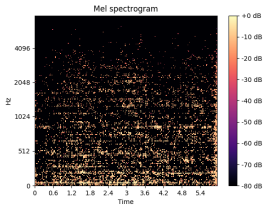


(b) Translation by U-Net

Figure 3: U-Net results



(a) Original Spectrogram



(b) Translation by SegNet

Figure 4: SegNet Results

Architecture	MSE	MAE
SegNet	0.07	0.1
U-Net	0.01	0.06

Table 1: Comparison of metrics at convergence

8 Possible Improvements

Through empirical obvservation we come to the conclusion that the model is not able to preserve the variablility of sparse pixels in an original input. So it straightaway gives the values for those pixels as 0 is the output. Which was not in the case of U-Net decoder, this evidence suggests that the amount of information passes from the encoder to the decoder is not enough for SegNet. So a naive solution would be to increase the latent space size, but this question the use of CNN Encoder-Decoder architecture which uses some kind of pattern/regularities present in the data. So we propose two methods. 1. Use RNN based patch wise reconstructing decoder, with it iterating on a fixed window of timesteps, but even this solution could struggle with the continuity across patches, one thing that could be done in this case is to use overlapping patches and then enforce soft-constraint on these and add it to the loss. 2. We can also use another latent space which captures the domain dependent information and use this information with the domain invariant information for reconstruction. Although it provides more information for reconstruction But we think

this would still not completely resolve the discrepancy seen in the sparse values. Both these models are heavy to train, either with increased time(RNN based solution) or increased number of parameters.