DCASE TASK 7: FOLEY SOUND SYNTHESIS

Technical Report

Ashwin Pillay[1], *Sage Betko*[1], *Ari Liloia*[1], *Hao Chen*[1], *Ankit Shah*[1]

[1] Carnegie Mellon University, Pittsburgh, USA, {apillay, sbetko, aliloia, haoc3, aps1}@andrew.cmu.edu

ABSTRACT

Foley sound synthesis refers to the creation of realistic, diegetic sound effects for a piece of media, such as film or radio. We propose building a deep learning system for Task 7 of the DCASE 2023 challenge that can generate original mono audio clips belonging to one of seven foley sound categories. Our training dataset consists of 4,850 sound clips from the UrbanSound8K, FSD50K, and BBC Sound Effects datasets. We aim to better the subjective and objective quality of generated sounds by passing as much meaningful information about the input data into latent representations as possible. The primary innovation in our submission is the change from using melspectrograms to using CEmbeddings (combined embeddings), which are input to the VQ-VAE and consist of melspectrograms concatenated with latent representations of audio produced by a pre-trained MERT model. Our submission to track A utilizes a pre-trained MERT model; as such, PixelSNAIL was trained on CEmbeddings. Our submission to track B utilizes PixelSNAIL retrained only on melspectrograms. Our code can be found here: `https://github.com/ankitshah009/foley-sound-synthesis_DCASE_2023`. Our submission for Track A is somewhat experimental and did not yield ideal results, but we feel it is worth documenting our work as MERT is a bleeding-edge model.

1. INTRODUCTION

Foley sound refers to diegetic, non-musical sound effects that convey the sounds produced by events depicted in a piece of media, such as radio or film. The process of creating complex sound environments from scratch is time-consuming and expensive; a method for convincingly synthesizing sounds could improve the content creation workflow. It could also be used to synthesize and augment other datasets. In this project, we build on the DCASE baseline model to create a model that generates original audio clips belonging to one of seven foley sound categories, namely *DogBark, Footstep, GunShot, Keyboard, MovingMotorVehicle, Rain*, and *Sneeze/Cough* [1].

2. MODEL DESCRIPTION

2.1. Optimizations to the baseline

While experimenting with the baseline, we observed that the default VQ-VAE learning rate was too large to yield any meaningful result, so we added a cyclic learning rate scheduler. Considering our time and compute constraints, we developed an optimized training scheme that could give us acceptable results within a days worth of training on a single, consumer-grade GPU. We accomplished this by implementing mixed precision training. We also ablated our batch size, reducing them to 16 for VQ-VAE and 8 for PixelSNAIL training. Lastly, we also implemented a system that employs the trained model on inference mode to return the FAD scores for 32 randomly-generated foleys of each of the aforementioned classes for subsequent evaluation.

2.2. Using CEmbeddings: An enhanced audio representation

As an alternative to the baseline model, we propose enhancing the melspectrogram input with higher-level audio features corresponding to factors like its key and acoustics. We believe such representations aid the model to utilize more domain-specific information while learning intra-class and inter-class qualities of the foleys. To this end, we integrated a pretrained encoder, MERT-v1-330M, as a preprocessor to our system.

MERT [2] [1] is a large-scale model trained on music audio for general music understanding. It has an architecture similar to HuBERT[3], a model for self-supervised speech representation learning that has been proven to capture higher-level acoustical features than melspectrograms. While HuBERT is trained on 16 kHz speech data, MERT has been specifically trained using a Masked Language Model (MLM) paradigm on 24 kHz music / audio data. The audio-specificity of MERT embeddings and its higher sampling rate results in more granular and meaningful representation of foley features than HuBERT embeddings. Moreover, MERT has been validated against a variety of music information retrieval (MIR) tasks like genre classification and key detection. The developers of MERT state that across the zeroth dimension of its embeddings, there is a gradual increase in the level of features, e.g. the first few dim0 features represent lower-level features like the time-frequency variations and the last few represent higher-level features like the key to which the piece of input audio belongs. While features like the key are more relevant to music than foleys, we believe the model could utilize this information to identify differences between foleys of the same class; for e.g. differences in the bark of a young Chihuahua and an adult Bulldog.

To aid concatenation of the melspectrograms with MERT embeddings, we modified how the former was obtained. This was done by increasing the mel frequency bands to 129 and increasing the hop size to 320 samples. We hypothesize that the increased features provided by MERT will compensate for the increase in melspectrogram hop size. Finally, we combined the two embeddings to form Combined Embeddings (**"CEmbed"**), as shown in Fig 1.

The use of CEmbed over plain melspectrograms required retraining all the downstream models in our system, along with significant changes to their architectures as described in the following

[1] MERT-v1-330M Huggingface: `https://huggingface.co/m-a-p/MERT-v1-330M`

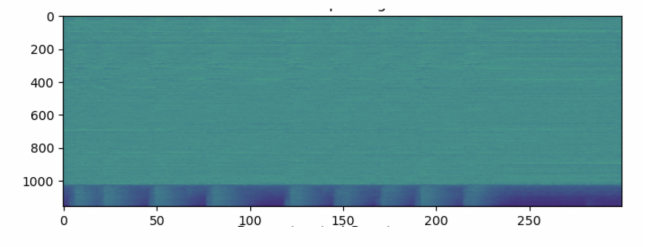

Figure 1: Plot of a CEmbed for one sample in the development set. The lowest and least uniform-looking rows represent the melspectrogram, while the upper rows are made up of the MERT-generated embeddings.

subsections. For a brief comparison of the changes made to the input embedding of the baseline and the final model, refer Table 1.

Table 1: Differences in sizes between analagous variables used in the baseline and final models. The melspectrogram sizes are (frequency band step, time step).

Variable	Baseline Shape	Final Shape
Audio Input	(22050 x 4, 1)	(24000 x 4, 1)
Melspectrogram	(80, 344)	(129, 300)
MERT Encodings	-	(1023, 300)
Input to VQVAE	(80, 344)	(1152, 300)
VQ-VAE Latent	(20, 80)	(288, 75)

2.3. Enhancements to VQ-VAE: MVQVAE

To optimize the latent representations of foleys generated from the incoming CEmbeds, we made several changes to the baseline VQ-VAE architecture. The resulting model is termed MERT - VQVAE (**MVQVAE**) with its main enhancements described as follows:

2.3.1. Foley Conditioning

The baseline VQ-VAE model learns an unconditional representation of sound, without any additional information about the category of sound during optimization or inference. Hence the responsibility of conditional sound generation lies solely with PixelSNAIL, which is tasked with learning to sample from the generalized codewords that make of the VQ-VAE's codebook, in order to assemble sequences based on the unique distribution of each sound category. However, the baseline VQ-VAE tends to produce codewords with similar conditional distributions across foley categories, which can make it difficult for PixelSNAIL to learn category-specific distributions. We hypothesize that this difficulty arises because the similar distributions cause PixelSNAIL to confuse categories, resulting in poor generation quality. To address this, we introduce a single linear layer that receives the average pre-quantization channel values of the latent representation and the foley category of the input. The cross-entropy loss between the predicted and the actual foley category is added to the total loss scaled by a factor of 1×10^{-2}.

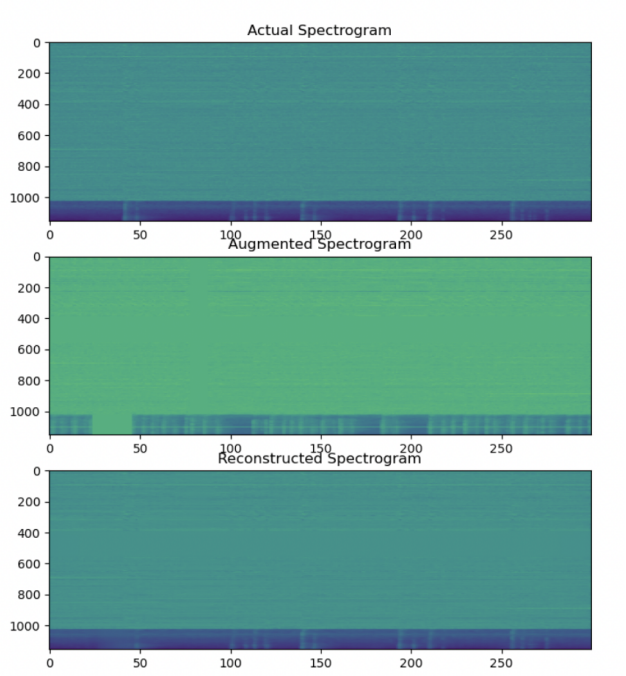

Figure 2: From top to bottom: the actual CEmbed, an example of an augmented training input to the model, and the reconstructed output of MVQVAE

2.3.2. CEmbed-specific model expansions

The CEmbeds in our new model are ≈ 14 times larger than the melspectrograms; the baseline VQVAE cannot operate on them as is. Thus, one key enhancement brought by MVQVAE include increasing the size of the dictionary maintaining the codebook vectors that can represent a single encoder output from 512 to 1024. Additionally, we added a parallel ResNet block in the encoder and decoder to increase its capacity to grasp the increased information provided by the CEmbeds. We included asynchronous time and frequency-masking data augmentations in the training paradigm to prevent the model from over-associating redundant relationships that may exist in the melspectrogram and the MERT embeddding of a given CEmbed. Fig. 2 demonstrates this training paradigm.

2.4. Optimizing PixelSNAIL for CEmbeds: Zen Mode

When applied to CEmbeds, the baseline version of PixelSNAIL suffers from impractical matrix multiplications. The scaled dot-product attention used in PixelSNAIL has an $O((TF)^2)$ memory requirement, where T and F are the time and feature dimensions of the quantized encodings from MVQVAE. This quadratic scaling makes self-attention impractical for longer sequence lengths, especially with the increased feature dimensionality introduced by MERT. Our group proposed an approach called **Zen Mode** to balance PixelSNAIL's efficiency with the preservation of CEmbeds' additional dimensionality.

Zen mode reduces the computation complexity of the self-attention mechanism in PixelSNAIL by incorporating trainable strided causal convolutional layers over the key and query vectors and transposed causal convolutions over the attention output. The

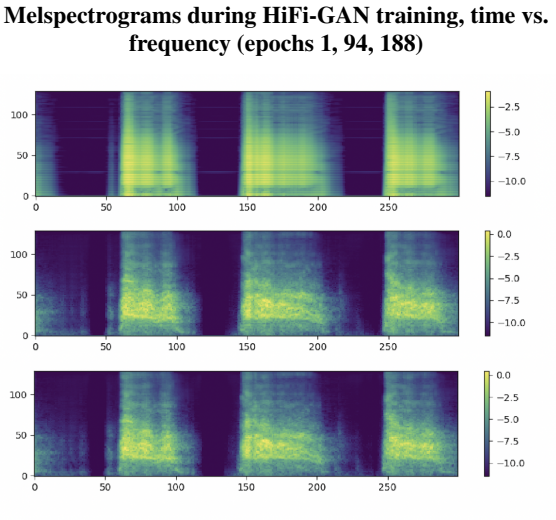

Figure 3: Frequency against time melspectrogram output of HiFi-GAN during training, at epochs 1, 94, and 188 (top to bottom) - over multiple epochs, the melspectrograms become more refined

convolutional layers downsample the input to the attention block, representing higher-level, coarser information from the embeddings and decreasing computational complexity. Our model applies a downsampling factor of 4, reducing the cost of computing the self-attention matrix by a factor of 16. Meanwhile, the actual CEmbed data is used without any downsampling. This allows us to model longer sequences while not sacrificing useful CEmbed feature data in PixelSNAIL's decoder hidden states.

Maintaining causality is essential for autoregressive models like PixelSNAIL. Standard transposed convolutions do not inherently possess causal properties, so we introduce a novel technique called *causal transposed convolution*. Causal transposed convolutions combine the upsampling capability of transposed convolutions with the causality property required for autoregressive modeling. This ensures that the generated output maintains causality.

To the best of our knowledge, the use of zen mode and causal transposed convolutions have not yet been proposed in the machine learning literature, making this a unique contribution of this work. With these enhancements, we term the new model as **Zen PixelSNAIL**.

2.5. Modifications to HiFi-GAN: MHiFiGAN

The pre-trained HiFi-GAN provided by DCASE expects VQVAE-decoded melspectrograms to generate audio at 22050 Hz. Since MVQVAE returns decoded CEmbeds, we propose MERT HiFi-GAN (**MHiFiGAN**), a model trained from scratch to vocode CEmbeds to audio at 24000 Hz. In contrast from HiFiGAN that performed dilation by a factor of 256, MHiFiGAN dilates incoming CEmbed, which have a feature rate of 75 Hz, by a factor of 320. This also accounts for errors in rounding the duration of the foley sounds to 4 seconds, an area in which the previous model was prone to error.

To make MHiFiGAN robust against imperfections in the MVQVAE-decoded CEmbeds, we modified the training paradigm of MHiFiGAN such that its trained on time and frequency masked CEmbeds.

2.6. Submissions to tracks A and B

Our submission to track A utilizes a pre-trained MERT model; as such, PixelSNAIL was trained on CEmbeds. Our submission to track B utilizes PixelSNAIL retrained only on melspectrograms.

3. PRELIMINARY RESULTS

The DCASE development dataset was split into a train and validation set for model evaluation. The train set consisted of 4360 samples and the validation set contained 245 samples. The validation set was constructed with a stratified random sample where 35 samples were randomly selected from each category, and the remaining samples were assigned for training.

3.1. Baseline Model

We have successfully implemented and trained the baseline solution described in [4], surpassing the results of the challenge organizers in all seven foley sound categories. The baseline model's FAD scores evaluated on the development dataset are provided in Table 2.

Table 2: FAD scores on DCASE development set (lower is better).

ID	Category	FAD (DCASE)	FAD (Ours)
0	DogBark	13.411	**8.958**
1	Footstep	8.109	**4.189**
2	GunShot	7.951	**6.765**
3	Keyboard	5.230	**3.086**
4	MovingMotorVehicle	16.108	**11.319**
5	Rain	13.337	**9.321**
6	Sneeze/Cough	3.770	**2.675**

Our training runs for VQ-VAE and PixelSNAIL are openly available to view on Weights & Biases.[2] We would also like to present a few example sounds generated by our current model in each category.[3]

Following the training procedure by [4], we trained the VQ-VAE for 800 epochs with a learning rate of 3×10^{-3}; we reduced the batch size to 16 from 64 in order to fit within a single GPU. We have exceeded the baseline performance with only 265 training epochs of PixelSNAIL, whereas [4] train for 1500 epochs. We attribute this primarily to our reduction in batch size from 32 to 8 and our addition of a cyclic learning rate scheduler with a reduced initial learning rate of 1×10^{-5}. Our use of PyTorch's automatic mixed-precision (AMP) training enabled us to complete the training of both the baseline VQ-VAE and PixelSNAIL models in under 24 hours on a single NVIDIA RTX A4000 with 16GB of VRAM.

3.2. Conditioned VQ-VAE and MVQVAE

Table 3 presents the results for the baseline and conditioned VQ-VAE models trained on Melspectrograms. Table 5 shows the same but for the MVQVAE. The addition of the classification loss term reduces both the train and validation MSE reconstruction loss.

[2] https://wandb.ai/audio-idl/Foley-sound-synthesis_DCASE_2023-baseline_dcase2023_task7_baseline

[3] Audio synthesis examples: https://drive.google.com/drive/folders/10LdqxEeVerVNEqcAb3uWjjpxnlmH27Jd

We see a significant reduction in latent loss, which measures the difference between the pre- and post-quantization encodings. The encoder output is mapped once to the codewords to obtain training data for PixelSNAIL, and then again to decode PixelSNAIL generation output during synthesis, so it is critical to obtain a low latent loss. This measures the degree of misalignment between the codebook and the encoder output, and hence the level of noise introduced by mapping between the encodings and codebook vectors.[4]

We hypothesize that the addition of class-conditioning described in section 2.3.1 while training the VQ-VAE/MVQVAE helps to better structure the latent space, as it allows the model to separate features unique to each sound category. This separation enables the codebook to hold more meaningful codewords that cater to individual sound categories, ultimately leading to a more effective use of the codebook's capacity.

Table 3: Loss terms in the baseline (unconditioned) and conditioned Melspectrogram based VQ-VAE.

Train

Model	MSE	Cross-Entropy	Latent Diff
Conditioned	**0.14056**	0.02053	**0.00167**
Baseline	0.14395	–	0.00183

Validation

Model	MSE	Cross-Entropy	Latent Diff
Conditioned	**0.14814**	0.02145	**0.00171**
Baseline	0.19166	–	0.00222

Table 4: Loss terms in the conditioned and unconditioned MVQ-VAE.

Train

Model	MSE	Cross-Entropy	Latent Diff
Conditioned	**0.357**	0.0859	**0.0179**
Unconditioned	0.4084	–	0.2973

Validation

Model	MSE	Cross-Entropy	Latent Diff
Conditioned	**0.2636**	0.02145	**0.0208**
Unconditioned	0.3196	–	0.3669

3.3. MHiFi-GAN

The baseline HiFiGAN model provided to us pre-trained, so we are unable to report its metrics to compare it with the results of MHiFi-GAN. However, through playback of the audio generated, we can validate that the model improves the quality of CEmbed to audio conversion over several epochs. Table 6 summarizes the validation and training metrics obtained for MHiFi-GAN after training it for 180 epochs.

4. OBSTACLES TO FINAL TRACK A RESULTS

Due to upstream modifications made to accommodate more detailed input representations, we had to enhance and train these models ourselves. One challenge we faced during training was requiring a fully trained MVQVAE to extract codes for Zen PixelSNAIL training.

Table 5: Training & Validation Metrics for MHiFiGAN.

Train		
Discriminator Loss	Generator Loss	Mel Recon. L1
3.041	27.911	0.3336
Validation		
Discriminator Loss	Generator Loss	Mel Recon. L1
2.961	27.760	0.3281

Despite implementing Zen mode optimizations, the increased size of Zen PixelSNAIL introduced numerous engineering challenges.

We experimented with several MVQVAE configurations. The first, MVQVAEv1, contained 512 codewords. To train Zen PixelSNAIL on MVQVAEv1 codes and include CEmbeds within our fixed compute budget of 16GB VRAM, we decreased its parameter count by reducing the number of channels from 256 to 128 and the number of residual blocks from 4 to 3. After several days, the model reached a saturation point at 50% accuracy and could not learn further, necessitating training from scratch on a larger model.

We discovered that the 512-codeword limitation of MVQ-VAEv1 hindered its ability to reconstruct CEmbeds. We trained a second model, MVQVAEv2, with 1024 codewords, which resulted in better reconstruction MSE and significantly improved qualitative reconstruction during listening tests on the HiFi-GAN waveform output. We restored the channel count and the number of residual blocks in Zen PixelSNAIL to their original values and trained on the larger MVQVAEv2 codes.

Our final configuration faced a multitude of engineering challenges as we attempted to scale up. The larger model was particularly susceptible to exploding gradients, which corrupted the optimizer state. Due to Zen PixelSNAIL's four serial decoder blocks, a significant accumulation of error occurred when applied to the larger CEmbeds. To stabilize training, we implemented gradient clipping and experimented with different values of the maximum gradient norm.

PixelSNAIL trained slower than expected on the larger MERT representations, despite renting four A40s to try to finish it in time. We prepared a track B submission using smaller discrete T-F representations that we knew PixelSNAIL could learn quickly so that we would have something not"outlyingly bad", in the event that our Track A samples were not ideal. While the samples produced by Track B are decent, we are disappointed by our Track A results, but are proud of the work we put in.

5. CONCLUSIONS

We create embeddings that represent both the lower-level time-frequency variances and the higher-level acoustical and musical features of the foleys. We then enhance our models to utilize this information for the intrinsic development of more detailed and distinguishable statistical distributions of each foley class.

We introduced potentially innovative techniques such as class conditioning to increase the inter-class distance between foleys, Zen Mode to streamline attention-context computations without sacrificing input quality, and Causal Transpose CNNs to support dilation in auto-regressive prediction problems.

Although we encountered issues when training our models, we believe our techniques warrant further investigation and we will be continuing our work after the competition deadline.

6. REFERENCES

[1] S. B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 28, no. 4, pp. 357–366, Aug. 1980.

[2] Y. Li, R. Yuan, G. Zhang, Y. Ma, C. Lin, X. Chen, A. Ragni, H. Yin, Z. Hu, H. He, *et al.*, "Large-scale pretrained model for self-supervised music audio representation learning," 2022.

[3] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," 2021.

[4] X. Liu, T. Iqbal, J. Zhao, Q. Huang, M. D. Plumbley, and W. Wang, "Conditional sound generation using neural discrete time-frequency representation learning," *2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, 2021.