Music Demixing Workshop

Transfer Learning with Jukebox for Music Source Separation

Wadhah Zai El Amri*[1], Oliver Tautz[1], Helge Ritter[1], and Andrew Melnik[1]

1 University of Bielefeld

Abstract

In this paper, we demonstrate a neural network architecture that uses representations of a publicly available pretrained *Jukebox* model and transfer learning to solve the problem of audio source separation. Jukebox takes 3 days of training on 256 GPUs. In this work, we demonstrate how to adapt Jukebox's audio representations for the problem of extracting an audio source from a single mixed audio channel. Results demonstrate an inspiring idea to tackle this challenge. Our approach is fast to train. We provide an open-source code implementation of our architecture.

Introduction

Music source separation from mixed audio is a challenging problem, especially if the source itself should be learned from a dataset of examples. Additionally, such models are expensive to train from scratch. We tested our model on the MUSDB18-HQ (Rafii et al., 2019b) dataset which supplies full songs with groundtruth stems of: bass, drums, vocals and other, which includes instruments such as guitars, synths etc. The task is to separate a mixed audio channel into the separately recorded instruments, called stems here. Most baseline models in the Music Demixing Challenge 2021 (Mitsufuji et al., 2021) used masking of input transformed to frequency domain by short-time Fourier transformation. *Demucs* (Défossez et al., 2019) showed a successful approach that works in waveform domain. *Demucs* is an autoencoder, based on a bidirectional long short-term memory network, with an architecture inspired by generative approaches. This encouraged us to adapt *Jukebox* (Dhariwal et al., 2020), a powerful, generative model using multiple, deep Vector Quantized-Variational Autoencoders (VQ-VAE) (Oord et al., 2017) to automatically generate real sounding music, and using its publicly available pretrained weights for the task.

Related Works

Transfer learning helped deep learning models reach new heights in many domains, such as natural language processing (Devlin et al., 2018),(Raffel et al., 2019) and computer vision (Han et al., 2018),(Gao & Mosalam, 2018). Although relatively unexplored for the audio domain, (Lim et al., 2016) proved feature representation learned on speech data could be used to classify sound events. Their results verify that cross-acoustic transfer learning performs significantly better than a baseline trained from scratch. TRILL (Shor et al., 2020) showed great results of pre-training deep learning models with an unsupervised task on a big dataset of speech samples. Its learned representations exceeded SOTA performance on a number of downstream tasks with datasets of limited size.

We take a similar approach that is heavily based on *Jukebox* (Dhariwal et al., 2020). It uses

*corresponding author

multiple VQ-VAEs to compress raw audio into discrete codes. They are trained self-supervised, on a large dataset of about 1.2 million songs, needing the compute power of 256 V100 to train in acceptable time. Our experiments show that *Jukebox's* learned representations can be used for the task of source separation.

Method

Architecture

Our architecture utilizes *Jukebox's* (Dhariwal et al., 2020) the standard variant of the publicly available pre-trained VQ-VAE model. *Jukebox's* uses three separated VQ-VAEs. We use only the smallest one with the strongest compression. It employs dilated 1-D convolutions in multiple residual blocks to find a less complex sequence representation of music. An audio sequence x_t gets mapped by an encoder E_1 to a latent space $e_t = E_1(x_t)$ of 64 dimensions so that it can be mapped to the closest prototype vector in a collection C of vectors called *codebook*. These 2048 prototype vectors, denoted c_{st}, are learned in training and help to form a high-quality representation.

The rate of compression for a sequence is called the hop length, for which a value of 8 is used. It depends on the stride values of the convolutional layers. We set the stride value to 2 as well as the down sampling to 3. All other values remain as defined in (Dhariwal et al., 2020). After mapping to the codebook, a decoder D aims to reconstruct the original sequence. In summary, equation (1)

$$y_t = D(argmin(\|E_1(x_t) - c)\|) \text{ for } c \in C \tag{1}$$

describes a full forward pass through the VQ-VAE, where $ y_t $ is the prediction for an input sequence x_t and $\|.\|$ is the euclidean norm. For further technical details on the used VQ-VAE architecture, refer to the paper of Dhariwal et al. (Dhariwal et al., 2020). The model is fine-tuned on data for one stem, learning good representations for a single instrument. In addition, we train a second encoder E_2, identical to the one already mentioned, to project an input sequence of the mixture to the space already known by the codebook and decoder. For deployment, the encoder of the VQ-VAE is switched with the second one, effectively mapping from the full mixture to one stem.

Data

Our models are trained on the MUSDB18-HQ (Rafii et al., 2019a) dataset, also used in the music demixing challenge (Mitsufuji et al., 2021). It consists of 150 full-length songs, sampled at 44KHz, providing the full audio mixture and four stems, vocals, bass, drums and other for each sample, which can be regarded as ground truth in the context of source separation. We train on the full train set composed of 100 songs, testing is done on the remaining 50.

Training

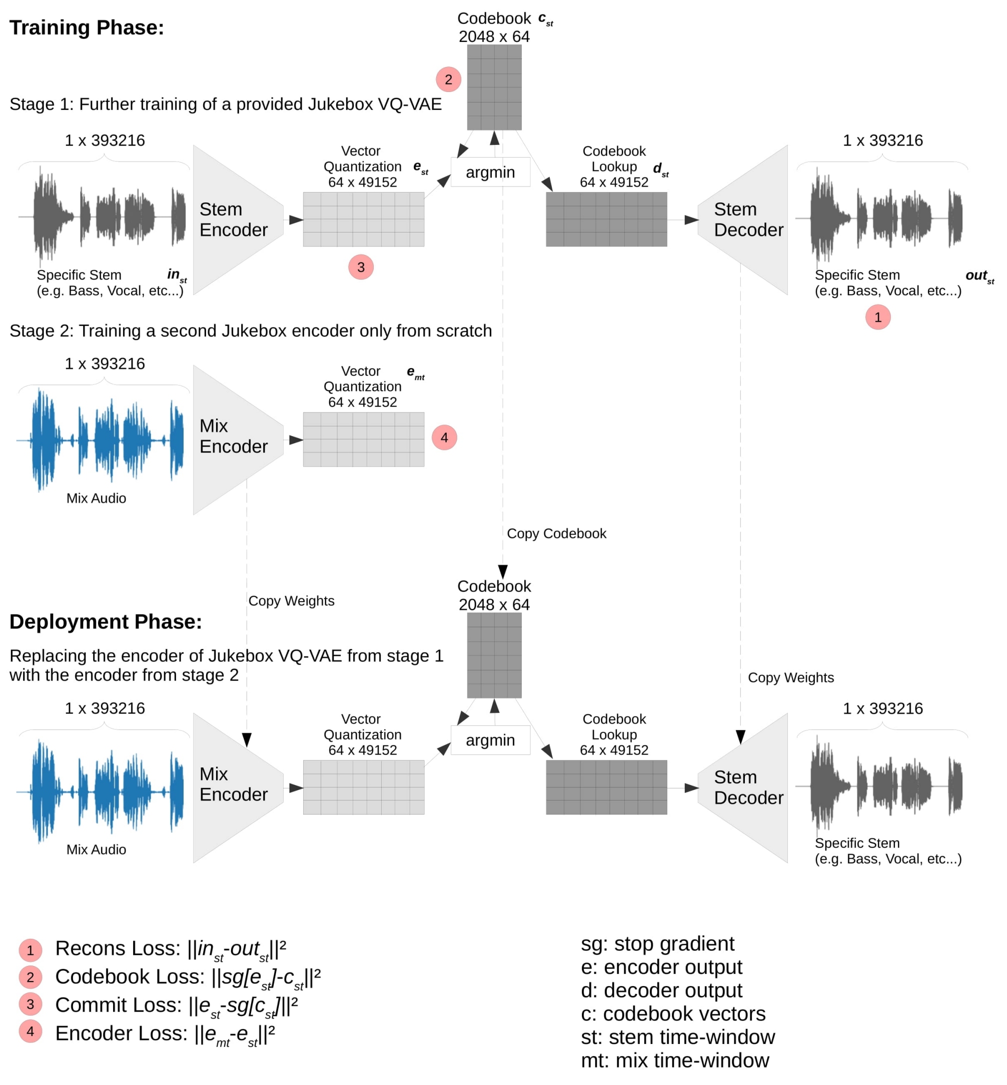


Figure 1: Visualization of the proposed transfer learning model architecture.

One model is trained per stem (see Fig. 1), furthermore, each is trained in two stages. In stage one, we train the adapted VQ-VAE (our Model 1) to produce good latent representations of a single stem specifically. *Jukebox's* provided weights are fine-tuned with a self-supervised learning task on the data for one stem with the same three losses, $L = L_{recons} + L_{codebook} + \beta L_{commit}$ used by (Dhariwal et al., 2020) so that the auto-encoder learns how to compress a single stem and reconstruct it.

For stage two, the second encoder is trained on the mix to learn the same encoding as the already trained encoder in the VQ-VAE. So for each training sample ($x_m t$: the sequence of the mixed audio, $x_s t$: the sequence of stem audio), we feed $x_s t$, to the already trained encoder E_1, producing e_{st}. Separately, the full mixture $x_m t$ is passed through the new encoder E_2, yielding e_{mt}. Now, we can backpropagate through E_2 using MSE loss $||e_{st} - e_{mt}||^2$. To clarify, we should mention that the weights of E_1 are not updated in stage 2. For deployment, we use the VQ-VAE trained in stage 1, but swap in the encoder trained in stage 2. On a more technical note, in both training stages and deployment, the data is processed chunk wise, with a size of about 9 seconds. For a clear overview of the content of this chapter refer to Figure

1.

For all conducted experiments that will be defined in the next section, two Tesla GPUs with 16Gb each are used. The length of each input sequence is equal to 393216 data points as used by *Jukebox*. The batch size is equal to 4.

To benchmark the conducted experiments, signal-to-distortion ratio (SDR) metric is used, which is a common metric in other SOTA papers(Défossez et al., 2019)(Stöter et al., 2019)(Hennequin et al., 2020)(Sawata et al., 2021)(Stoller et al., 2018). 'Total' SDR is the mean SDR for all stems.

Experiments and Results

The main key point of this paper consists of proving that it is possible to get decent audio quality by using transfer learning. For this, we did two different experiments on the four audio stems. We trained the first VQ-VAE networks for each audio stem from scratch without using any pretraining values. Then, we trained in a second experiment, the VQ-VAE with pretrained weights of the *Jukebox*. For all these VQ-VAE we pick the checkpoint 80K and train the corresponding encoder of the second model. For all the second models (experiment 1 and 2) we initialized randomly the weights, when starting the training.

For the first experiment, we found out that all the results are low, and no good audio quality is reached. The SDR values are equal or near 0 for all the four stems.

For the second experiment, the model converges after 32 hours of training in total on two Tesla GPU units with 16GB of VRAM each.

We present the results in the following figure 2, corresponding each to the SDR results of the second experiment for the four audio stems.

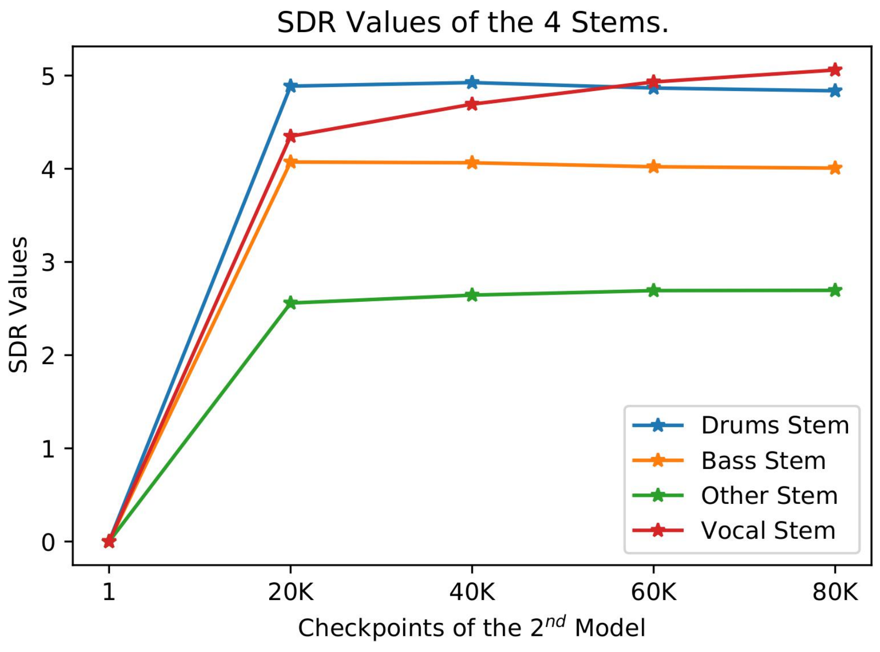
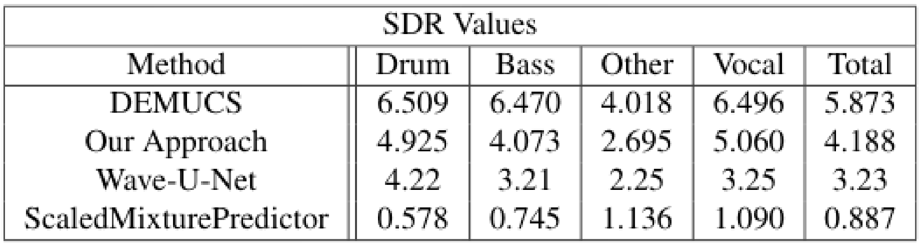


Figure 2: SDR results of the 4 audio signal stems for the second experiment.

In Figure 2 demonstrates decent SDR values for networks trained with a pretraining weights in comparison to others trained with random initialized weights from scratch. It is also to be deduced that it is even enough to train until early checkpoint values, such as 20K, in order to get fairly good SDR values. Then, the checkpoint 20K is reached after 16 hours for each of the two models on two Tesla GPUs. Table (1) gives a comparison of different approaches for audio signal separation. Our approach achieves here comparable results, when benchmarked with other state-of-the-art networks.



SDR Values					
Method	Drum	Bass	Other	Vocal	Total
DEMUCS	6.509	6.470	4.018	6.496	5.873
Our Approach	4.925	4.073	2.695	5.060	4.188
Wave-U-Net	4.22	3.21	2.25	3.25	3.23
ScaledMixturePredictor	0.578	0.745	1.136	1.090	0.887

Figure 3: Table 1: SDR values for different approaches for the four stems.

Conclusion

Transfer learning is used in modern architectures for image processing, neural language processing, etc. In this work we demonstrate how to use transfer learning for the problem of audio signal processing and in particular in demixing audio signal from a single mixed audio channel into four different stems: drums, bass, vocals and rest. We show that it is possible to succeed in such tasks with a small-sized dataset and this is achieved, when using pretrained weights from *Jukebox* (Dhariwal et al., 2020) network.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, *abs/1810.04805*. http://arxiv.org/abs/1810.04805

Défossez, A., Usunier, N., Bottou, L., & Bach, F. R. (2019). Demucs: Deep extractor for music sources with extra unlabeled data remixed. *CoRR*, *abs/1909.01174*. http://arxiv.org/abs/1909.01174

Dhariwal, P., Jun, H., Payne, C., Kim, J. W., Radford, A., & Sutskever, I. (2020). *Jukebox: A generative model for music*. http://arxiv.org/abs/2005.00341

Gao, Y., & Mosalam, K. M. (2018). Deep transfer learning for image-based structural damage recognition. *Computer-Aided Civil and Infrastructure Engineering*, *33*(9), 748–768. https://doi.org/https://doi.org/10.1111/mice.12363

Han, D., Liu, Q., & Fan, W. (2018). A new image classification method using CNN transfer learning and web data augmentation. *Expert Systems with Applications*, *95*, 43–56. https://doi.org/https://doi.org/10.1016/j.eswa.2017.11.028

Hennequin, R., Khlif, A., Voituret, F., & Moussallam, M. (2020). Spleeter: A fast and efficient music source separation tool with pre-trained models. *Journal of Open Source Software*, *5*(50), 2154. https://doi.org/10.21105/joss.02154

Lim, H., Kim, M. J., & Kim, H. (2016). Cross-acoustic transfer learning for sound event classification. *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2504–2508. https://doi.org/10.1109/ICASSP.2016.7472128

Mitsufuji, Y., Fabbro, G., Uhlich, S., & Stöter, F.-R. (2021). *Music demixing challenge at ISMIR 2021*. http://arxiv.org/abs/2108.13559

Oord, A. van den, Vinyals, O., & Kavukcuoglu, K. (2017). Neural discrete representation learning. *CoRR*, *abs/1711.00937*. http://arxiv.org/abs/1711.00937

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, *abs/1910.10683*. http://arxiv.org/abs/1910.10683

Rafii, Z., Liutkus, A., Stöter, F.-R., Mimilakis, S. I., & Bittner, R. (2019a). *MUSDB18-HQ - an uncompressed version of MUSDB18*. https://doi.org/10.5281/zenodo.3338373

Rafii, Z., Liutkus, A., Stöter, F.-R., Mimilakis, S. I., & Bittner, R. (2019b). *MUSDB18-HQ - an uncompressed version of MUSDB18*. https://doi.org/10.5281/zenodo.3338373

Sawata, R., Uhlich, S., Takahashi, S., & Mitsufuji, Y. (2021). *All for one and one for all: Improving music separation by bridging networks*. http://arxiv.org/abs/2010.04228

Shor, J., Jansen, A., Maor, R., Lang, O., Tuval, O., Quitry, F. de C., Tagliasacchi, M., Shavitt, I., Emanuel, D., & Haviv, Y. (2020). Towards learning a universal non-semantic representation of speech. *Interspeech 2020*. https://doi.org/10.21437/interspeech.2020-1242

Stoller, D., Ewert, S., & Dixon, S. (2018). *Wave-u-net: A multi-scale neural network for end-to-end audio source separation*. http://arxiv.org/abs/1806.03185

Stöter, F.-R., Uhlich, S., Liutkus, A., & Mitsufuji, Y. (2019). Open-unmix - a reference implementation for music source separation. *Journal of Open Source Software*, *4*(41), 1667. https://doi.org/10.21105/joss.01667