STATIONARY/TRANSIENT AUDIO SEPARATION USING CONVOLUTIONAL AUTOENCODERS

Gerard Roma
CeReNeM
University of Huddersfield
Huddersfield, UK
g.roma@hud.ac.uk

Owen Green
CeReNeM
University of Huddersfield
Huddersfield, UK
o.green@hud.ac.uk

Pierre Alexandre Tremblay
CeReNeM
University of Huddersfield
Huddersfield, UK
p.a.tremblay@hud.ac.uk

ABSTRACT

Extraction of stationary and transient components from audio has many potential applications to audio effects for audio content production. In this paper we explore stationary/transient separation using convolutional autoencoders. We propose two novel unsupervised algorithms for individual and and joint separation. We describe our implementation and show examples. Our results show promise for the use of convolutional autoencoders in the extraction of sparse components from audio spectrograms, particularly using monophonic sounds.

1. INTRODUCTION

The problem of identifying transients in audio signals (especially musical audio) has received significant attention in the phase vocoder literature, given the difficulties posed by transients to sinusoidal models. As a consequence, a number of sines + transients + noise models were proposed [1, 2]. Transient and stationary components can in fact be related with general signal models prevalent in audio effects [3].

These models are often applied to monophonic sounds, but their application to broad polyphonic signals remains challenging. Meanwhile, researchers focusing on separation of polyphonic signals into their component sources have developed a similar separation task, often dubbed harmonic-percussive source separation (HPSS). This name obviously assumes the presence of harmonic and percussive components in audio. However, techniques employed for this task often do not actually take into account harmonicity of musical tones and instead focus on other aspects of typically harmonic components of polyphonic signals. Most algorithms are based, in one way or another, on the observation that percussive and harmonic components tend to form straight vertical and horizontal lines in the spectrogram. This property can be called the *anisotropic smoothness* [4]. Several works have been developed to exploit this using non-negative factorization algorithms [5, 6]. A very popular approach is to simply use a combination of two median filters [7].

Separation of audio into stationary and transient components, that is, without modeling sinusoids, was proposed in a recent study [8]. This perspective allows the application of ideas based on anisotropic smoothness to digital audio effects. In this sense, this task remains in an abstract domain related to signal models, which makes quantitative evaluation elusive.

In this paper, we propose two algorithms for transient/stationary separation using convolutional autoencoders (CAE). Autoencoders are neural network algorithms that purposely realize imperfect replicas of input signals based on some constraints. Thanks to current neural network programming libraries, such constraints can be specified directly into cost functions, without having to worry about their derivative. This provides a promising framework for experimenting with digital audio effects. In this paper we explore their use for transient/stationary separation by implementing anisotropic smoothness constraints from the HPSS literature in the cost functions.

2. CONVOLUTIONAL AUTOENCODERS

Autoencoders are neural network algorithms that try to reconstruct the input from a typically lower dimension hidden representation. The encoder is typically the combination of an affine transform with weights W and biases b with some non-linear activation σ:

$$h = \sigma(Wx + b). \tag{1}$$

Here, h is a hidden representation of x with dimensionality determined by the weight matrix. The decoder then performs the inverse operation to obtain a reconstruction y:

$$y = \sigma(W'h + b'). \tag{2}$$

This is typically accomplished using some variant of stochastic gradient descent (SGD) that learns the parameters W, b, W' and b' to minimize the some distance metric between x and y. Autoencoders have been extensively used in machine learning, usually not for the reconstruction itself but for learning useful features from data. The parameters that produce the hidden representation h are then used in other neural networks for e.g. image classification. In order to avoid that the algorithm learns to exactly copy the input to the output, which would not yeld useful features, the main strategies are choosing a lower dimensionality for h or adding some sparsity constraint to the cost function.

An analogy of traditional autoencoders with non-negative factorization (NMF) algorithms used for audio separation was proposed in [9] but evaluated only for the supervised case. Supervised neural networks used in separation of musical audio [10] can be seen as supervised autoencoders, in the sense that the output is the same shape as the input.

Traditional autoencoders, however, process data in one dimension and thus cannot be used to learn time-frequency patterns. The usual solution of stacking several spectral frames quickly degenerates into prohibitive computational costs.

Convolutional neural networks (CNNs) have become the standard algorithm for image classification and object recognition. They have also been shown to work for speech recognition [11] and audio classification [12]. In CNNs, the weights are typically square convolution kernels that are shared, i.e. each kernel is convolved with the whole image. The resulting representation is downsampled with respect to the input image size (which can be further downsampled with pooling operations), but typically composed by multiple channels corresponding to each learnable kernel.

Convolutional autoencoders (CAEs) arose in this context, allowing the use of 2D convolution operations for learning features. In a CAE, the operation in Equation 1 is rewritten as :

$$h^i = \sigma(X * W^i + b^i), \tag{3}$$

where $*$ represents a 2D convolution operation. Here the input X is a matrix. The hidden representation is now a tensor, $h \in \mathbb{R}^{d,m,n}$, with d determined by the number of kernels (hence the index i for each convolution). In addition to the size of the input and the kernel, dimensions m and n can be affected by several parameters of the convolution, such as input padding and stride. Thus, h can be in all a higher-dimensional representation than the input, but the information has to be transmitted through the convolution with small (typically 5x5) kernels.

While these are conventional convolution layers used in CNNs, the particularity of CAEs is to introduce an upsampling convolution that allows restoring the original size in the decoder:

$$Y = \sigma(h^i * W'^i + b'^i). \tag{4}$$

This operation is informally called "deconvolution" [13], or more technically *fractionally strided convolution* [14], and it involves padding and re-shaping the kernels into a convolution matrix of a size that allows recovering the original size through convolution with the hidden representation. One interesting property of this architecture is that it can be used for images (in our case magnitude spectrograms) of arbitrary size.

Supervised networks with deconvolution decoders have recently started to appear in the source separation literature [15, 16]. In this paper, we explore the use of this architecture in an unsupervised setting for stationary / transient separation of audio. It is common in AEs and CAEs to implement restrictions in the loss function, in addition to the output being similar to the input. This feature can thus be used to devise new audio effects. In the case of CAEs, the loss function can take into account both the time and frequency dimensions and promote vertical or horizontal lines as commonly done for HPSS.

3. TRANSIENT / STATIONARY AUDIO SEPARATION

In this section we describe different loss functions that can be used to train a CAE. As noted, our approach consists of using the network to process a magnitude spectrogram. We define X to be such spectrogram (e.g. it has been obtained from some complex spectrogram C), and assume it to be a sum of two components:

$$X = X_t + X_s. \tag{5}$$

Here X_t represents the time-frequency bins associated with transients, and X_s the ones associated with stationary components. We regard this as a useful abstraction and not as a physical mixture, beyond the fact that musical sounds typically contain transients and steady tones. It is often useful to distinguish a noise component that is not associated with transients. While we do not model this component directly, we observe in Section 3.1 that a basic CAE can be used to remove background noise. In Section 3.2 we show a model that can be used to individually estimate X_t or X_s. This allows using a different time-frequency grid that may be more appropriate for each situation. In this case, a corresponding complex estimate can be recovered using the original phase, e.g. for a complex STFT:

$$\hat{C} = \hat{X} e^{\phi j}, \tag{6}$$

where $\hat{X}$ can either be $\hat{X}_t$ or $\hat{X}_s$, and ϕ is the phase of the original spectrogram.

On the other hand, for ensuring that X is recovered by the sum of both estimates, it may be convenient to estimate a soft mask, i.e:

$$M_t = \frac{\hat{X}_t}{\hat{X}_t + \hat{X}_s}, \tag{7}$$

$$M_s = \frac{\hat{X}_s}{\hat{X}_t + \hat{X}_s}, \tag{8}$$

using a common transform for both components. $\hat{C}$ is then obtained as $M_t \otimes C$ or $M_s \otimes C$, where $\otimes$ denotes the element-wise product. In Section 3.3 we describe a model for jointly obtaining $\hat{X}_t$ and $\hat{X}_s$ from the same spectrogram.

3.1. Basic CAE

A basic CAE implementation simply tries to recover the input. A suitable loss function would then be the mean square error (MSE) between the input X and the output Y:

$$L_{MSE} = \frac{1}{TF} \sum (X - Y)^2, \tag{9}$$

where T and F are the dimensions of the spectrogram. The goal of the algorithm is then to find an optimal set of kernels that allow this reconstruction through 2D convolutions. Using this function implies the danger of simply copying the input. It is easy to see that a convolution kernel with a single active weight would accomplish that. One common solution is to add a sparsity constraint on the hidden representation. However, here we are interested in the output (i.e. transient or stationary components) being sparser than the input. Promoting a sparse hidden representation does not directly accomplish that, because the decoder can try to learn to re-create the (non-sparse) input from the sparse hidden representation. Hence, we add a sparse penalty to the output directly:

$$L = L_{MSE} + \lambda_1 ||Y||_1, \tag{10}$$

where $||*||_1$ denotes the L1 norm. The parameter λ_1 then controls the sparsity of the output spectrogram. In early experiments with this model, we observed it may have interesting applications to denoising and dereverberation. In this sense, traditional autoencoders have been applied to speech enhancement [17]. It can also be used to implement more experimental effects. However our main goal in this work is the transient/stationary decomposition.

3.2. Individual extraction

Estimation of transient or stationary components from the input signal can be promoted by adding more terms to the loss function. We regard the difference across either the time or the frequency axis as a cost for estimating transient or stationary components respectively:

$$d_f = \frac{\sum_{t,f} (Y(t,f) - Y(t,f-1))^2}{||Y||_2{}^2}, \tag{11}$$

$$d_t = \frac{\sum_{t,f}(Y(t,f) - Y(t-1,f))^2}{||Y||_2{}^2}, \quad (12)$$

where $|| * ||_2$ denotes the L2 norm, and t and f are time and frequency indices. The loss for estimating either the transient or the stationary components is then computed by adding respectively $\frac{d_f}{d_t+\varepsilon}$ or $\frac{d_t}{d_f+\varepsilon}$ (where ε is a small number to prevent division by 0) to L_{MSE}:

$$L_S = L_{MSE} + \lambda_1||Y||_1 + \lambda_2\frac{d_f}{d_t+\epsilon}, \quad (13)$$

$$L_T = L_{MSE} + \lambda_1||Y||_1 + \lambda_2\frac{d_t}{d_f+\epsilon}, \quad (14)$$

Parameters λ_1 and λ_2 can here be mapped to user interface parameters: the first one defines the level of sparsity (i.e. how much magnitude will be lost in the process) and the second biases it towards the desired component.

3.3. Joint extraction

Estimating both transient and stationary components simultaneously has the potential advantage of allowing a more discriminative model that can use the input data to provide two estimates. The estimates can then be used to construct time-frequency masks as described in Equations 8 and 7. Here, the output of the autoencoder is a tensor $Y \in \mathbb{R}^{2,M,N}$ where M and N correspond to the original spectrogram size. For simplicity of notation, we denote $Y_t \in \mathbb{R}^{T,F}$ and $Y_s \in \mathbb{R}^{T,F}$ as the outputs of the CAE for transient and stationary components respectively. The MSE loss then needs to be rewritten as:

$$L_{MSE} = \frac{1}{TF}\sum(X - (Y_t + Y_s))^2. \quad (15)$$

The terms df and d_t can now be computed separately for Y_t and Ys respectively. The loss function for the CAE is then:

$$L_{ST} = L_{MSE} + \lambda_1||Y||_1 + \lambda_2\frac{d_{t1}}{d_{f1}+\epsilon} + \lambda_3\frac{d_{f2}}{d_{t2}+\epsilon}, \quad (16)$$

where d_{t1} / d_{f1} are computed from Y_s as in Equations 11 and 12, and d_{t2} / d_{f2} are equally computed from Y_t .

4. IMPLEMENTATION

In order to test the proposed approach, we implemented the CAE models described in Sections 3.1, 3.2 and 3.3, respectively denoted here as $cae1$, $cae2$ and $cae3$. The implementation was based on the pytorch library.[1] Figure 1 shows the layout that is common to the three models. We used 5x5 convolution kernels, which are widely used for images and have also been used for audio classification [12]. All networks were devised with 4 convolution kernels and one single hidden representation of 4 channels. Inputs to all convolutions were padded with 2 bins on each side and dimension. This means there was really no downsampling neither pooling, and the hidden representations had the same dimension of the input, which helped recovering the fine details of the input. Both the encoder and the decoder used Rectified Linear Units (ReLU) as activation functions. Initialization for weights connected to ReLUs is conventionally implemented as specified in [18]. However, we found that for this unsupervised setting, results could be unstable due to random initialization. On one hand, different initial balances between the components of the loss function could lead the network to fall into a local minimum. On the other, the network could end in a slightly different state for the same number of iterations even when converging to a stable solution. In order to make the networks predictable, we used a basic CAE trained to optimize only L_{MSE} to pre-initialize the weights. The proposed models were then used to *fine tune* the weights with the additional loss components. This had the side effect of choosing a random seed. It has been shown that pre-training is robust to changes in the random seed [19]. We verified that, for different pre-trained networks, our models would always converge to a stable solution. However, thinking about the use of the algorithm in an interactive effects processor, the predictability resulting from the use of a fixed random seed was also beneficial. All models were trained using the ADAM [20] variant of stochastic gradient descent (SGD). Like in [9], each spectrogram was used as a single batch, both for pre-training and fine tuning. For the pre-training, we used two different datasets, one composed of monophonic loops and one with polyphonic music signals. For dealing with monophonic sounds, the pre-training dataset was obtained by randomly sampling 100 loop sounds from the collection bundled with Apple's Logic Pro software. For dealing with polyphonic mixtures, the pre-training dataset was created by extracting one minute from each song in the test set (50 songs) of the DSD100 dataset.[2] For both the training and pre-training stages, the $weight\_decay$ parameter, available in pytorch, was used. This corresponds to an l_2 regularization in the weights, which is omitted in the formulation for clarity. A value of 0.01 was used for $cae1$ and $cae2$, while for $cae3$ a higher value of 0.5 helped prevent the weights getting biased towards one of the two outputs. All networks were trained for 100 epochs. Spectrograms were computed using 20 ms windows with 15 ms overlap except when noted. The code for the implementation can be obtained from `https://github.com/flucoma/DAFX-2018`.

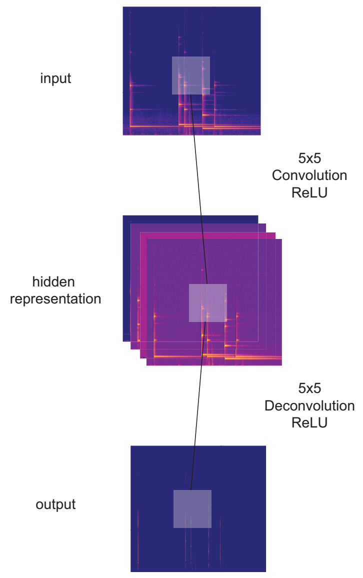
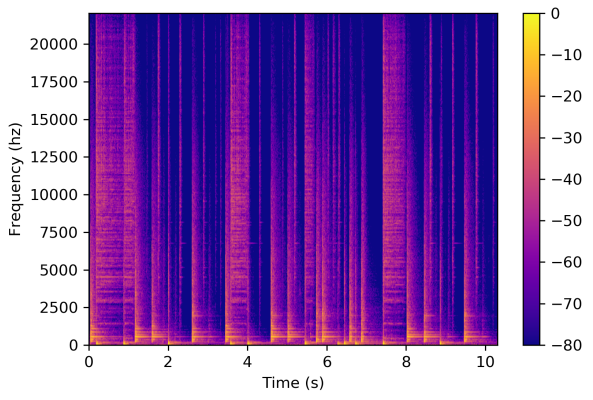


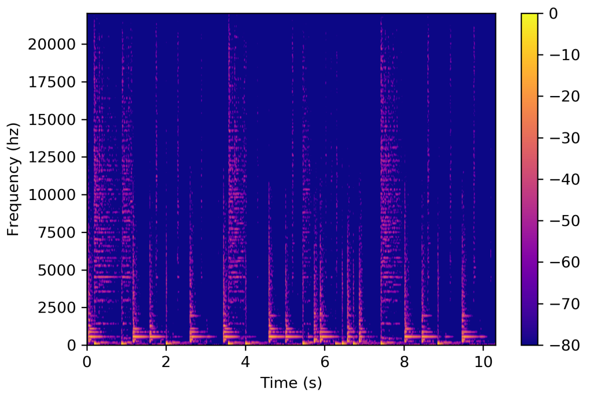
Figure 1: Convolutional autoencoder network structure

[1] `http://pytorch.org/`

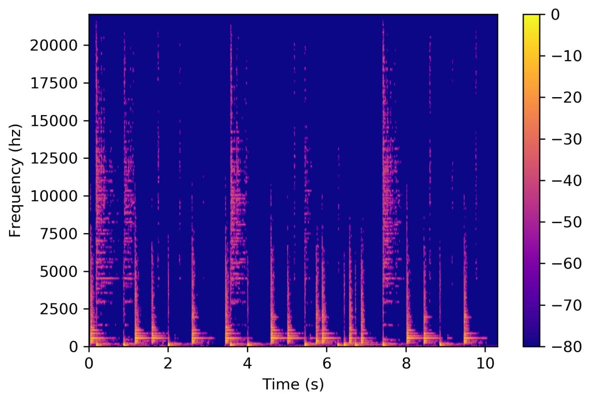
[2] `https://github.com/faroit/dsdtools`



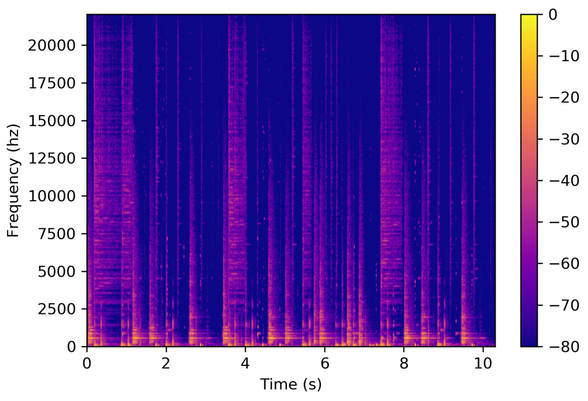
(a) Original drum loop sound



(b) Drum loop processed by cae1



(c) Drum loop with thresholded magnitude



(d) Drum loop remixed with cae1 variations

Figure 2: Using cae 1 on a drum loop

5. EXAMPLE RESULTS

In this section we show examples of the use of $cae1$, $cae2$ and $cae3$ as described in the previous sections. We first show a creative application of $cae1$ with a drum loop, then we analyze the separation into steady and transient components of $cae2$ and $cae3$ using a monophonic and a poyphonic sample. All audio examples can be listened in the companion web page for this paper: `http://www.flucoma.org/DAFX-2018/`.

The original drum loop is shown in Figure 2a. A sparse version obtained with $cae1$ is shown in Figure 2b. A lot of the resonance of the drums has been lost. For comparison, Figure 2c shows a version of the original with the same number of zero entries (around 94%) as the processed version (i.e. magnitude bins were sorted and zeroed below a threshold to obtain the same number of zeros). It seems that $cae1$ focuses more on the harmonics of the drums. We found this effect can be used for creative processing to obtain multiple variations of the same sample. As an example, Figure 2d shows an example where multiple copies using different values of λ_1 at different window and hop sizes have been mixed with the original.

We now focus on transient/stationary separation using $cae2$ and $cae3$. Figure 3 is a monophonic fragment of a glockenspiel melody from Freesound.org[3]. The original sound includes significant background noise. Figures 4a and 4b show the magnitude spectrograms of the separation with $cae2$. The background noise has been eliminated, and the stationary and transient components are clearly separated. When listening to the sounds, it can be noted

[3]`https://freesound.org/people/bbatv/sounds/332932/`

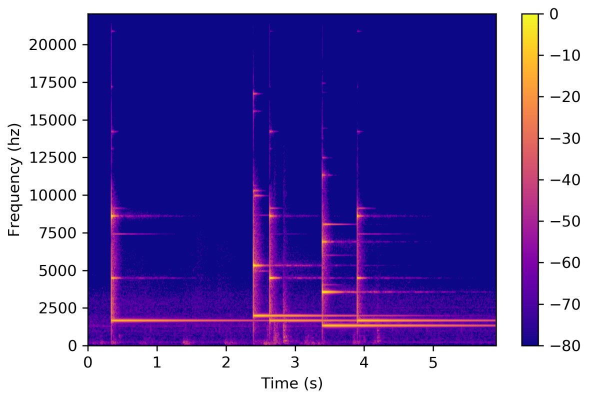
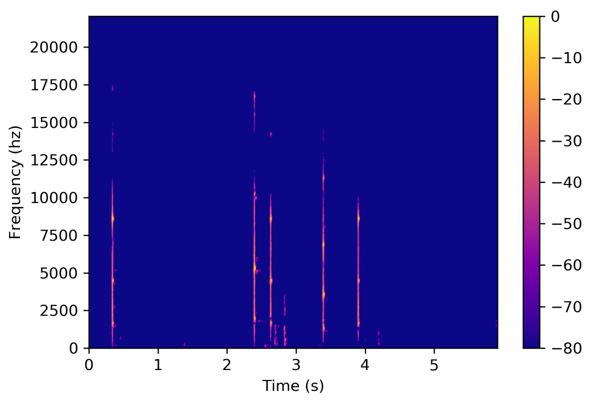
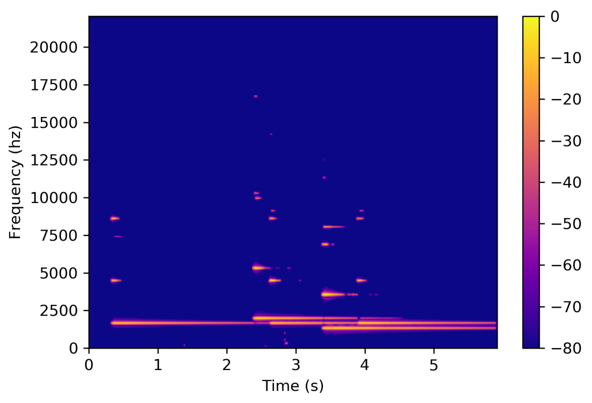


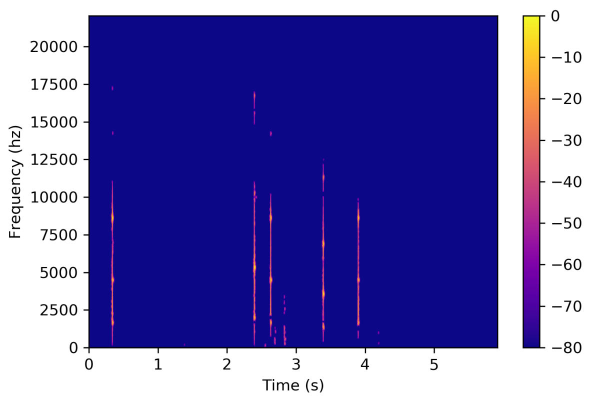
Figure 3: Original glockenspiel sound



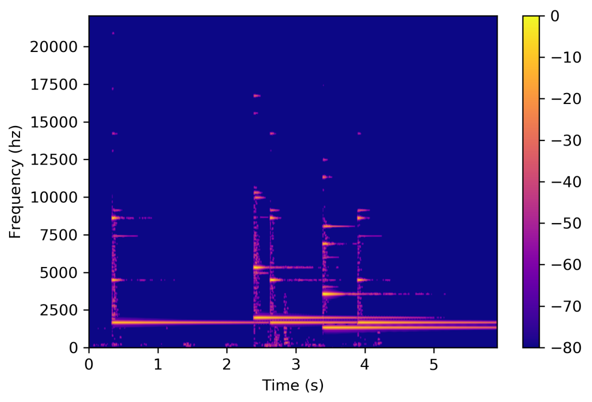
(a) Separation of glockenspiel transients with $cae2$.



(b) Separation of glockenspiel stationary components with $cae2$.



(c) Separation of glockenspiel transients with $cae3$.



(d) Separation of glockenspiel stationary components with $cae3$.

Figure 4: Using cae2 and cae3 on a glockenspiel sound

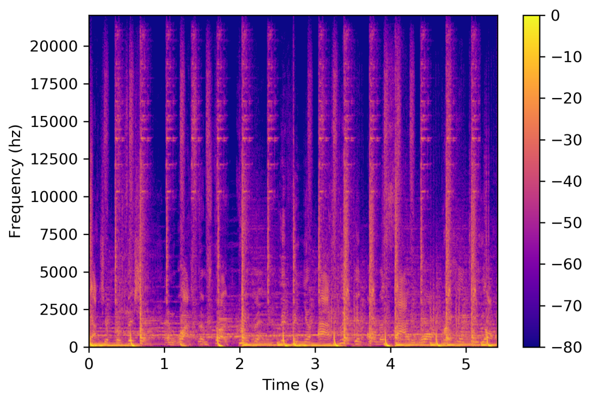


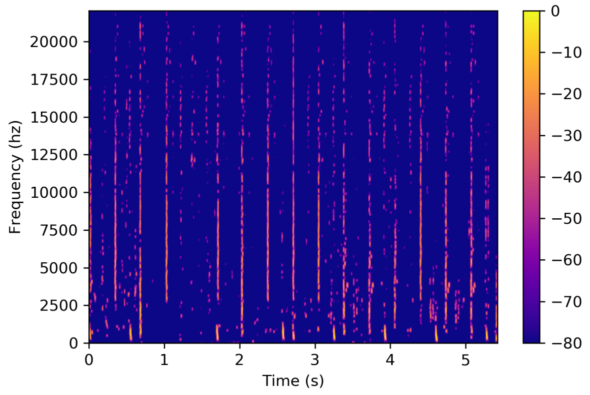
Figure 5: Original polyphonic mixture

that the transients still retain some of the pitch information but the duration is very short. In the stationary components, the attack has been clearly removed. The parameter values for the transient estimation were $\lambda_1 = 8\text{e-}4$, $\lambda_2 = 300$. For the stationary estimation, the values were $\lambda_1 = 4\text{e-}5$, $\lambda_2 = 10$. Figures 4c and 4d show the results with $cae3$. The spectrograms look also sparse, but the stationary components seem to show a stronger attack, which can be attributed to the use of the soft mask. When listening to the audio it can be noted that the attack is in fact very soft. In this case, the parameters were tuned to $\lambda_1 = 5\text{e-}5$, $\lambda_2 = 0.2$, $\lambda_3 = 0.3$.

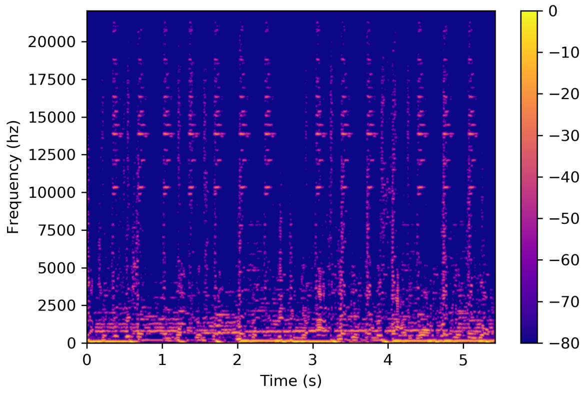
For both models, the strategy was to set first the target level of sparsity with λ_1 and then adjust the rest of parameters. However, we noted that the competition of both estimates in $cae3$ makes it more difficult to find appropriate values for the parameters.

Figures 5, 6a, 6b, 6c and 6d correspond to a hip hop music excerpt[4]. The separation is obviously more difficult. For both networks, the separation of transients produces noticeable musical noise. They are still good indicators of the downbeat of the rhythm. The stationary components in $cae2$ are biased towards the bass, which is salient and perhaps the only instrument producing steady tones. Contrastingly for $cae3$ the stationary part is remarkably more simlar to the mix, but with smoothed transients, which could be attributed to the joint estimation. The parameters for $cae2$ were $\lambda_1 = 1\text{e-}4$, $\lambda_2 = 100$ and $\lambda_1 = 4\text{e-}5$, $\lambda_2 = 50$ for transient and stationary components, and $\lambda_1 = 1\text{e-}4$, $\lambda_2 = 2$, $\lambda_3 = 6$ for the joint model $cae3$.

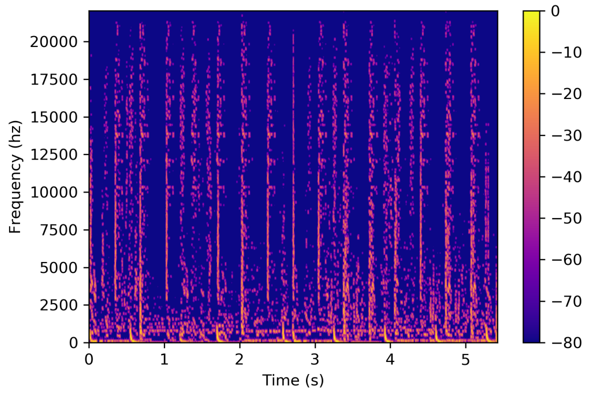
[4]The excerpt was extracted from *Attention* by Catburglaz, `http://catburglaz.com`



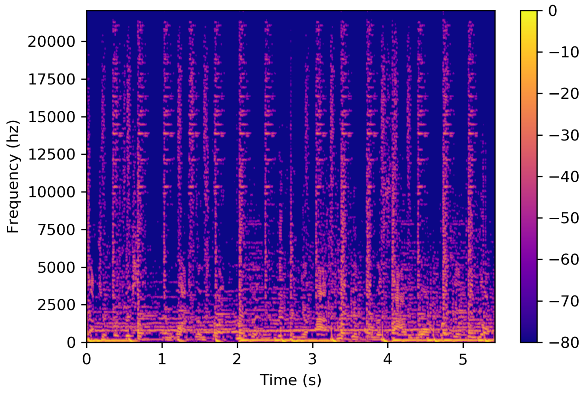
(a) Separation of transients in polyphonic mixture with $cae2$.



(b) Separation of stationary components in polyphonic mixture with $cae2$.



(c) Separation of transients in polyphonic mixture with $cae3$.



(d) Separation of stationary components in polyphonic mixture with $cae3$.

Figure 6: Using cae2 and cae3 with a polyphonic mixture

6. CONCLUSIONS

In this paper, we have explored the use of unsupervised convolutional autoencoders for audio transformation in the time-frequency domain. Specifically, we have shown that by programming custom loss functions they can be tuned to separate stationary and transient components. The results are encouraging, especially for monophonic sounds, while polyphonic mixtures are still challenging. One interesting aspect of this work is the possibility to control the learning process, producing different levels of sparseness and different qualities of transients and stationary components. This brings more flexibilty than HPSS approaches such as median filtering. Such flexibility is of particular interest to us as it presents opportunities for creative exploration: being able to tune processors by ear to fit aesthetically with the materials and the context in which they are used is a very important aspect of artistic interfaces. For future work, we plan to work on more useful mappings of the loss functions to user interface parameters.

7. ACKNOWLEDGEMENT

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement n. 725899).

8. REFERENCES

[1] Scott N Levine and Julius O Smith III, "A sines+ transients+ noise audio representation for data compression and time/pitch scale modifications," in *Audio Engineering Society Convention 105*. Audio Engineering Society, 1998.

[2] Tony S Verma, Scott N Levine, and Teresa HY Meng, "Transient modeling synthesis: a flexible analysis/synthesis tool for transient signals.," in *Proceedings of the International Computer Music Conference (ICMC)*, 1997.

[3] Axel Roebel, "Between physics and perception: Signal models for high level audio processing," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2010.

[4] H. Tachibana, N. Ono, H. Kameoka, and S. Sagayama, "Harmonic/percussive sound separation based on anisotropic smoothness of spectrograms," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 12, pp. 2059–2073, Dec 2014.

[5] T. Virtanen, "Monaural sound source separation by non-negative matrix factorization with temporal continuity and sparseness criteria," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 3, pp. 1066–1074, March 2007.

[6] Francisco Jesus Canadas-Quesada, Pedro Vera-Candeas, Nicolas Ruiz-Reyes, Julio Carabias-Orti, and Pablo Cabanas-Molero, "Percussive/harmonic sound separation by non-negative matrix factorization with smoothness/sparseness constraints," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2014, no. 1, pp. 26, 2014.

[7] Derry Fitzgerald, "Harmonic/percussive separation using median filtering," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2010.

[8] Kai Siedenburg and Simon Doclo, "Iterative structured shrinkage algorithms for stationary/transient audio separation," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2017.

[9] Paris Smaragdis and Shrikant Venkataramani, "A neural network alternative to non-negative audio models," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 86–90.

[10] Aditya Arie Nugraha, Antoine Liutkus, and Emmanuel Vincent, "Multichannel music separation with deep neural networks," in *24th European Signal Processing Conference (EUSIPCO)*. IEEE, 2016, pp. 1748–1752.

[11] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014.

[12] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al., "Cnn architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 131–135.

[13] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus, "Deconvolutional networks," in *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2010, pp. 2528–2535.

[14] Vincent Dumoulin and Francesco Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[15] Emad M Grais and Mark D Plumbley, "Single channel audio source separation using convolutional denoising autoencoders," in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2017, pp. 1265–1269.

[16] Andreas Jansson, Eric J. Humphrey, Nicola Montecchio, Rachel Bittner, Aparna Kumar, and Tillman Weyde, "Singing voice separation with deep U-Net convolutional networks," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2017, pp. 323–332.

[17] Xugang Lu, Yu Tsao, Shigeki Matsuda, and Chiori Hori, "Speech enhancement based on deep denoising autoencoder.," in *Interspeech*, 2013, pp. 436–440.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 1026–1034.

[19] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio, "Why does unsupervised pre-training help deep learning?," *Journal of Machine Learning Research*, vol. 11, no. Feb, pp. 625–660, 2010.

[20] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.