Unsupervised source separation with deep priors

Maurice Frank

July 13, 2020

Contents

Method 5

Modeling the priors 7

Sampling from the posteriors 8

Modeling the posteriors 9

Difficulties of this method 11

Method

In this chapter, we introduce the theoretical idea of our probability modelling of musical source separation. First, we explicitly state our chosen model of the problem, next derive a suitable optimization objective from it and then explain how we can optimize towards that.

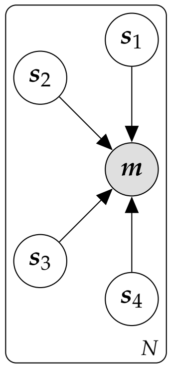

Figure 1: The used graphical model for the source separation task. We have the latent source channel variables $\boldsymbol{s}_k$. Exemplary here, as in our data, we have four sources. The mix $\boldsymbol{m}$ is observed.

We propose the graphical model as shown in Figure 1 as the generative story of music tracks being generated from separate source tracks. For each source, a sample is taken from the latent source distribution. The observed mix is generated deterministically from the full set of sources. Without loss of generality, we fix this function to be the mean.

Our stated task in **??** is to retreive the sources $\{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N\}$ from a given mix $\boldsymbol{m}$. Our model is trained without using sample tuples

$$(\boldsymbol{m}, \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N\}) : f(\{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N\}) = \boldsymbol{m}$$

which would show the relation between separate sources and mixes. The general idea is visualized in Figure 2.

Looking back at the graphical model in Figure 1, it implies the following factorization:

$$\begin{aligned} p(\boldsymbol{m}) &= \int^N p(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N, \boldsymbol{m})\, d^N \boldsymbol{s} && (1) \\ &= \int^N p(\boldsymbol{m}|\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N) \cdot p(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N)\, d^N \boldsymbol{s} && (2) \\ &= \int^N p(\boldsymbol{m}|\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N) \cdot p(\boldsymbol{s}_1) \cdots p(\boldsymbol{s}_N)\, d^N \boldsymbol{s} && (3) \end{aligned}$$

While the conditional $p(\boldsymbol{m}|\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N)$ is not even probabilistic, as the mix is generated deterministically with the mean of the sources, the model posterior $p(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N|\boldsymbol{m})$ is intractable and precisely what we are interested in. Again we want to make it clear that this setup changes the typical optimization target, as seen in other source separation approaches. We factorize the distribution $p(\boldsymbol{m})$ of mixed songs, only then to extract the most likely *latent* components that generate this mix, the sources. Therefore we are explicitly modelling the generative process of the mixed songs, and only implicitly the separation task.

Already with the graphical model we introduce a fairly big as-

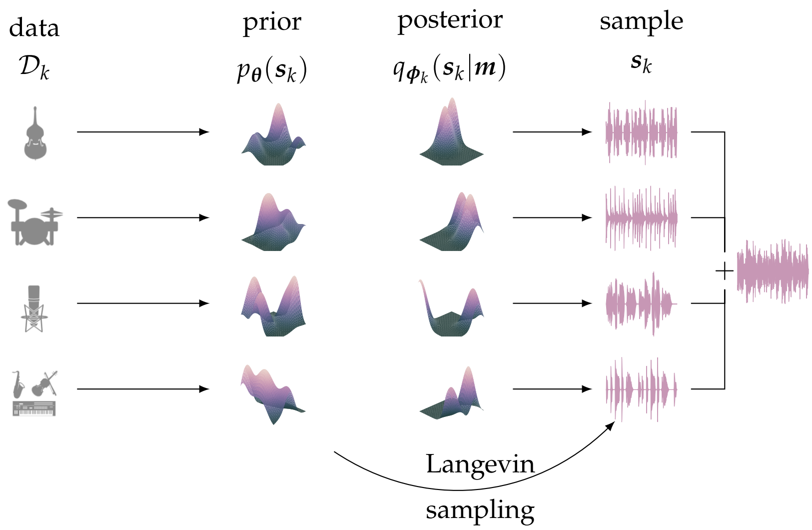

Figure 2: The method visualized

sumption, namely that we can model the source distributions independently from each other when modeling the joint:

$$p(\boldsymbol{m}, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_N) \equiv p(\boldsymbol{m}|\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N) \cdot \prod_k^N p(\boldsymbol{s}_k) \tag{4}$$

Intuitively this assumption does not, in general, hold for the musical domain. We can expect a different behaviour for a *guitar* in a song involving a second source like a *percussion set*. The joint of those two models is different than their independent densities [1]. Nevertheless, this assumption is crucial to be able to model the prior instrumental models without needing the tuples $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N)$ of co-occurring stems.

[1] A practical counter-example: If learning the behaviour of drums, only from solo drum recordings, one will encounter complex rhythms and sounds. In many rock genres drums often exhibit simpler sounds, providing beat and tempo. These two distributions are not the same.

The general process is as follows:

First, because we assumed in our graphical model the latent sources to be independent, there exists a probability distribution $p(\boldsymbol{s}_k)$ for each source k. Thereout we need to choose a model which gives the parametrized approximated prior $p_{\boldsymbol{\theta}}(\boldsymbol{s}_k)$ with the parameters $\boldsymbol{\theta}$ which we optimize with the samples from $\mathcal{D}_k$.

Second, for each source, there exists a posterior given a mix $p(\boldsymbol{s}_k|\boldsymbol{m})$ from which we want to draw samples. Here two approaches are possible. Either we can propose an approximate posterior $q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})$, which is trained to minimize the divergence from the previously trained and fixed corresponding prior (VAE setting). Or, we can sample directly from the posterior using Stochastic Gradient Langevin Dynamics (SGLD)[2] without explicitly approximating the posterior distribution (sampling setting). Both optimization, either training or sampling, happen under the mixing constraint:

[2] Welling and Teh, "Bayesian Learning via Stochastic Gradient Langevin Dynamics", 2011.

$$\sum_{k=1}^{N} \boldsymbol{s}_k = \boldsymbol{m} \tag{5}$$

When using the VAE setting we then can sample from each posterior to retrieve a set of (approximately) correct source tracks. In the case of using Langevin dynamics, the iterative sampling will directly give this set from the prior distributions.

Thinking in the common terms of the VAE we need to choose:

1. a parametrization $p_{\boldsymbol{\theta}}(\boldsymbol{s}_k)$ of the *prior* distribution
2. a parametrized approximate posterior distribution $q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})$ with parameters $\boldsymbol{\eta}$
3. a parametrized *encoder* which gives the parameters for the posterior distribution $\texttt{Encoder}_{\boldsymbol{\phi}}(\boldsymbol{m}) = \boldsymbol{\eta}$
4. a *decoder* which returns the input $\boldsymbol{m}$ from the latents $\texttt{Decoder}(\{s_1, \ldots, s_N\}) = \boldsymbol{m}$

As stated before the decoder in our case is the mean function, thus not probabilistic and without parameters. It is certainly possible to parametrize the decoder with trainable parameters. This would imply though that the latent samples are not in the *direct form* of the source sounds but under an (unknown) transformation.

Modeling the priors

The first step in the training procedure is the training of the independent source priors $p(\boldsymbol{s}_k)$. We have to choose a model for estimating the density that results in smooth, tight densities but also is capable enough to capture the complex data space of audio data. We choose to estimate the priors with flow models for which we gave an overview in **??**. For different representations of the input, different network architectures are more suited. We experiment both with using directly the time-domain wave but also modelling on top of a spectral transformation of the time-domain data.

The two main variants of normalizing flows we build our priors from are the RealNVP and Glow. Their most important difference is the different formulation of the coupling layers. For the coupling layers, we have to split the layer input into two equally sized subsets, one of whom will be transformed using an invertible transformation parametrized by the other. The RealNVP masks the data spatially or over the channels. When done spatially a checkerboard binary mask is used (same for all channels), for the channel variant the channels are assigned in an even-odd switching. Glow, on the other hand, learns a 1×1-convolution which permutes the data channel-wise and than just splits the data along the channel dimension into two. For both RealNVP and Glow prior work exists specifically adapting the architecture to the time-series domain by integrating WaveNet modules, FloWaveNet[3] and WaveGlow[4], respectively.

Remove Glow if no experiment will use it

For the case of the time-domain representation, the data has only one channel (mono-audio). Therefore a Glow-like shuffling

[3] Kim et al., "FloWaveNet : A Generative Flow for Raw Audio", 2019.

[4] Prenger et al., "WaveGlow: A Flow-Based Generative Network for Speech Synthesis", 2018.

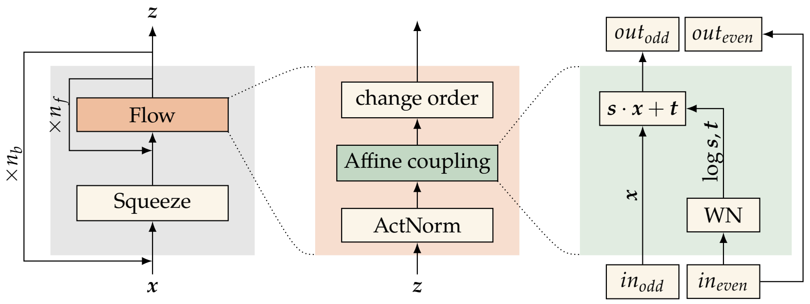

Figure 3: The building blocks for the prior model. The model consists of n_b blocks (left). Each block consists of n_f flows (middle). In each flow we apply activation normalization, followed by the affine coupling (right), after which the binary mask for the even/odd mapping is inverted. The affine coupling layer uses a WaveNet with the *even* set as the input to output scaling $\log s$ and translation t which which the *odd* set is transformed.

of channels is not possible and we resort to using spatial masking for the coupling layers. A one-dimensional mask is used over the time dimension. In the case of using a spectral representation of the audio data, the input samples have a lower time resolution but instead the frequency distribution over a deeper channel dimension. Therefore we can also experiment with channel-shuffling.

We make it clear that the success of this method depends strongly on the expressiveness and smoothness of the learned prior models. In the case of learning the posterior, variational learning will fit the posterior distribution under the prior distribution. If the prior does not contain enough variation of the actual data distribution, the de-mixing model cannot fit a conditional posterior similar to the sample in the mixed song. In the case of the sampling approach, this constraint is even more stressed. The mixed is generated by iteratively sampling from the set of prior distributions, any possible source sample, therefore, has to be found on the manifold of the prior. If any of the priors does not capture the full variations of the data distributions, neither method will be able to extract the correct separated sources.

More practically we again point out the high difficulty of modelling audio data. As laid out in **??** audio consists of information at greatly different resolutions. A generative model of an instrument has to model the low-range components, e.g. pitch, timbre, tremolo and high-range components, e.g. notes and rhythm.

Prior architecture

See the visualization of the prior model's architecture in Figure 3. We construct normalizing flow models, following the description in Kim et al.[5].

[5] Kim et al., "FloWaveNet : A Generative Flow for Raw Audio", 2019.

Algorithm 1: Training of the prior models

1: **for** $x \in \mathcal{D}_k$ **do**
2: **end for**

Sampling from the posteriors

After having estimated the N prior distributions the first possibility for retrieving sample estimates $\mathbf{s}_k$ is sampling from the posterior using Langevin dynamics[6]. With Stochastic Gradient Langevin Dynamics (SGLD) we can sample from $\boldsymbol{s}_i \sim p(\cdot|\boldsymbol{m})$ without computing, explicitly, the posterior $p(\boldsymbol{s}_i|\boldsymbol{m})$. For a overview of SGLD see **??**.

[6] Welling and Teh, "Bayesian Learning via Stochastic Gradient Langevin Dynamics", 2011.

Starting with the update step in SGLD we reformulate the update with our problem constraint:

$$\begin{aligned}
\boldsymbol{s}_k^{(t+1)} &= \boldsymbol{s}_k^{(t+1)} + \eta \cdot \nabla_{\boldsymbol{s}_k} \log p(\boldsymbol{s}_k^{(t)}|\boldsymbol{m}) + 2\sqrt{\eta}\epsilon_t && (6)\\
&= \boldsymbol{s}_k^{(t+1)} + \eta \cdot \nabla_{\boldsymbol{s}_k} [\log p(\boldsymbol{m}|\boldsymbol{s}_k) + \log p(\boldsymbol{s}_k) - \log p(\boldsymbol{m})] + 2\sqrt{\eta}\epsilon_t && (7)\\
&= \boldsymbol{s}_k^{(t+1)} + \eta \cdot \nabla_{\boldsymbol{s}_k} [\log p(\boldsymbol{m}|\boldsymbol{s}_k) + \log p(\boldsymbol{s}_k)] + 2\sqrt{\eta}\epsilon_t && (8)\\
&= \boldsymbol{s}_k^{(t+1)} + \eta \cdot \left[\|\boldsymbol{m} - \frac{1}{N}\sum_j^N \boldsymbol{s}_j^{(t)}\| + \nabla_{\boldsymbol{s}_k} \log p(\boldsymbol{s}_k) \right] + 2\sqrt{\eta}\epsilon_t && (9)
\end{aligned}$$

$\log p(\boldsymbol{m})$ is independent from $\boldsymbol{s}_k$ therefore no gradient.

η is the update step size and $\epsilon_t \sim \mathcal{N}(0, \mathbb{1})$.

Note that the final formulation of the update step is simply a constrained greedy hill-climb under the prior model with added Gaussian noise. Intuitively the artificially noised update makes it harder for the greedy optimization to get stuck in local minima of the prior surface.

Algorithm 2: The Langevin sampling procedure for source separation is fairly straight forward. For a fixed number of steps T we sample we take a step into the direction of the gradient under the priors and the gradient of the mixing constraint while adding Gaussian noise ϵ_t.

```
for t = 1...T do
    for k = 1...N do
        ε_t ~ N(0, 1)
        Δs_k^t ← s^t + η · ∇ log p(s^t) + 2√η ε_t
    end for
    for k = 1...N do
        s_k^{t+1} ← Δs_k^t − (η/σ²) · [m − (1/N) Σ_i^N s_i^t]
    end for
end for
```

The idea of using SGLD in combination with deep parametrized priors was, concurrently to this work, introduced in Jayaram and Thickstun[7]. The authors empirically show that

[7] Jayaram and Thickstun, "Source Separation with Deep Generative Priors", 2020.

$$\mathbb{E}\left[\|\nabla_{\boldsymbol{s}} \log p_\sigma(\boldsymbol{s})\|^2\right] \propto 1/\sigma^2$$

They argue that this surprising proportionality is stemming from the severe non-smoothness of the prior model. If the prior model, in the extreme case, exhibits a Dirac delta peak as the probability mass, then the estimation of the gradient with added Gaussian noise will it-self be proportional to the Gaussian. From there the authors argue, that the prior models have to be trained with noised samples, where the added noised is proportional to later used estimator noise.

Modeling the posteriors

Instead of formulating a sampling regime for approaching $\boldsymbol{s}_k$ we can also use the prior models to variationally train a model to estimate the parameters of an approximate posterior $q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})$. While estimating the true posterior $p(\boldsymbol{s}_k|\boldsymbol{m})$ is intractable, we choose a certain parametrized distribution as an approximation of the posterior and optimize the parameters to align with the true one.

We follow the same steps as previously shown generally for the latent variable models. First, we introduce an approximate posterior $q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})$ for each source channel. Next, we express the mix density as the expectation over those posteriors:

$$\log p(\boldsymbol{m}) = \mathbb{E}^N_{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}\left[\log p(\boldsymbol{m})\right] \tag{10}$$

From here we introduce the approximate posteriors in the expectation as before, just now for N separate priors:

$$\mathbb{E}^N_{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}\left[\log p(\boldsymbol{m})\right] = \mathbb{E}^N_{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}\left[\log \frac{p(\boldsymbol{m}, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_N)}{p(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N|\boldsymbol{m})}\right] \tag{11}$$

$$= \mathbb{E}^N_{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}\left[\log \frac{p(\boldsymbol{m}|\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N) \cdot \prod_k^N p(\boldsymbol{s}_k)}{\prod_k^N q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})} + \log \frac{\prod_k^N q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}{p(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N|\boldsymbol{m})}\right] \tag{12}$$

Note that again we use the assumption in Equation (4) to factorize the joint. This assumption is not being used for the independent approximate posteriors. While the true posterior certainly is the posterior of the joint source model, we choose our approximate posteriors to be independent. The expectation in (10) over those is still correct. The error arising from the independence assumption is captured in the thightness of the ELBO.

We arrive at the ELBO by leaving out the divergence of the approximate posterior to the true posterior:

$$\mathbb{E}^N_{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}\left[\log p(\boldsymbol{m})\right] \geq \sum_k^N \mathbb{E}_{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}\left[\log \frac{p(\boldsymbol{s}_k)}{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}\right] + \mathbb{E}_{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}\left[p(\boldsymbol{m}|\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N)\right] \tag{13}$$

The lower bound, as in the normal VAE, will be our optimization target for training the inference models that give the variational parameters $\boldsymbol{\phi}_k$. We can also formulate the objective for just one source, as the Encoders are trained independently[8]. Thus we come to:

[8] While the optimization of the Encoders is independent, the training is, of course, concurrent as the mixing conditions depends on all N source samples.

$$\mathbb{E}_{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}\left[\log p(\boldsymbol{m})\right] \geq \mathbb{E}_{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}\left[\log \frac{p(\boldsymbol{s}_k)}{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}\right] + \mathbb{E}_{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}\left[p(\boldsymbol{m}|\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N)\right] \tag{14}$$

$$= \mathbb{E}_{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}\left[\log \frac{p(\boldsymbol{s}_k)}{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}\right] + \|\boldsymbol{m} - \frac{1}{N}\sum_j^N \boldsymbol{s}_j\| \tag{15}$$

The Kullback-Leibler divergence term is computationally expensive, therefore the divergence is the stochastic estimate using just one mini-batch.

With Equation (15) we have the optimization objective set up and now have to choose a fitting parametrized posterior for $q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})$. In many recent

Algorithm 3: Training of the approximate posterior.

for $x \in \mathcal{D}_k$ **do**
end for

Difficulties of this method

1. Good prior densities

Bibliography

Burgess, Christopher P., Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner (2018). "Understanding Disentangling in Beta-VAE". In: arXiv: 1804.03599 [cs, stat].

Chorowski, Jan, Ron J. Weiss, Samy Bengio, and Aäron van den Oord (2019). "Unsupervised Speech Representation Learning Using WaveNet Autoencoders". In: *IEEE/ACM Trans. Audio Speech Lang. Process.* 27.12, pp. 2041–2053. arXiv: 1901.08810.

Cohen-Hadria, Alice, Axel Roebel, and Geoffroy Peeters (2019). "Improving Singing Voice Separation Using Deep U-Net and Wave-U-Net with Data Augmentation". In: arXiv: 1903.01415 [cs, eess].

Dauphin, Yann N., Angela Fan, Michael Auli, and David Grangier (2017). "Language Modeling with Gated Convolutional Networks". In: arXiv: 1612.08083 [cs].

Defossez, Alexandre, Neil Zeghidour, Nicolas Usunier, Leon Bottou, and Francis Bach (2018). "SING: Symbol-to-Instrument Neural Generator". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., pp. 9041–9051.

Défossez, Alexandre, Nicolas Usunier, Léon Bottou, and Francis Bach (2019). "Demucs: Deep Extractor for Music Sources with Extra Unlabeled Data Remixed". In: arXiv: 1909.01174 [cs, eess, stat].

Dinh, Laurent, David Krueger, and Yoshua Bengio (2015). "NICE: Non-Linear Independent Components Estimation". In: arXiv: 1410.8516 [cs].

Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio (2017). "Density Estimation Using Real NVP". In: arXiv: 1605.08803 [cs, stat].

Grais, Emad M., Dominic Ward, and Mark D. Plumbley (2018). "Raw Multi-Channel Audio Source Separation Using Multi-Resolution Convolutional Auto-Encoders". In: arXiv: 1803.00702 [cs].

Higgins, Irina, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner (2016). "Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In:

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Computation* 9.8, pp. 1735–1780.

Holschneider, M., R. Kronland-Martinet, J. Morlet, and Ph. Tchamitchian (1990). "A Real-Time Algorithm for Signal Analysis with the Help of the Wavelet Transform". In: *Wavelets*. Ed. by Jean-Michel Combes, Alexander Grossmann, and Philippe Tchamitchian. Inverse Problems and Theoretical Imaging. Berlin, Heidelberg: Springer, pp. 286–297.

ISO/TC 43 Acoustics (1975). *ISO 16 Acoustics Standard Tuning Frequency*.

Jansson, Andreas, Eric J. Humphrey, Nicola Montecchio, Rachel M. Bittner, Aparna Kumar, and Tillman Weyde (2017). "Singing Voice Separation with Deep U-Net Convolutional Networks". In: *ISMIR*.

Jayaram, Vivek and John Thickstun (2020). "Source Separation with Deep Generative Priors". In: arXiv: 2002.07942 [cs, stat] (cit. on p. 9).
Jordan, Michael I., Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul (1999). "An Introduction to Variational Methods for Graphical Models". In: *Machine Learning* 37.2, pp. 183–233.
Kalchbrenner, Nal, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aäron van den Oord, Sander Dieleman, and Koray Kavukcuoglu (2018). "Efficient Neural Audio Synthesis". In: arXiv: 1802.08435 [cs, eess].
Kaspersen, Esbern Torgard (2019). "HydraNet: A Network For Singing Voice Separation". In:
Kim, Sungwon, Sang-gil Lee, Jongyoon Song, Jaehyeon Kim, and Sungroh Yoon (2019). "FloWaveNet : A Generative Flow for Raw Audio". In: arXiv: 1811.02155 [cs, eess] (cit. on pp. 7, 8).
Kingma, Diederik P. and Prafulla Dhariwal (2018). "Glow: Generative Flow with Invertible 1x1 Convolutions". In: arXiv: 1807.03039 [cs, stat].
Kingma, Diederik P. and Max Welling (2014). "Auto-Encoding Variational Bayes". In: arXiv: 1312.6114 [cs, stat].
Kingma, Diederik P. and Max Welling (2019). "An Introduction to Variational Autoencoders". In: arXiv: 1906.02691 [cs, stat].
Kotelnikov, Vladimir (1933). "On the Carrying Capacity of the Ether and Wire in Telecommunications". In: *Material for the First All-Union Conference on Questions of Communication*. Moscow: Izd. Red. Upr. Svyazi RKKA.
Lluís, Francesc, Jordi Pons, and Xavier Serra (2019). "End-to-End Music Source Separation: Is It Possible in the Waveform Domain?" In: arXiv: 1810.12187 [cs, eess].
Narayanaswamy, Vivek Sivaraman, Sameeksha Katoch, Jayaraman J. Thiagarajan, Huan Song, and Andreas Spanias (2019). "Audio Source Separation via Multi-Scale Learning with Dilated Dense U-Nets". In: arXiv: 1904.04161 [cs, eess, stat].
Native Instruments (n.d.). *Stem Audio Format*.
Neal, Radford M. (2012). "MCMC Using Hamiltonian Dynamics". In: arXiv: 1206.1901 [physics, stat].
Paine, Tom Le, Pooya Khorrami, Shiyu Chang, Yang Zhang, Prajit Ramachandran, Mark A. Hasegawa-Johnson, and Thomas S. Huang (2016). "Fast Wavenet Generation Algorithm". In: arXiv: 1611.09482 [cs].
Prenger, Ryan, Rafael Valle, and Bryan Catanzaro (2018). "WaveGlow: A Flow-Based Generative Network for Speech Synthesis". In: arXiv: 1811.00002 [cs, eess, stat] (cit. on p. 7).
Pulse Code Modulation (PCM) of Voice Frequencies (1972). G.711. ITU-T.
Rafii, Zafar, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner (2017). *MUSDB18 - a Corpus for Music Separation*. Version 1.0.0.
Rethage, Dario, Jordi Pons, and Xavier Serra (2018). "A Wavenet for Speech Denoising". In: arXiv: 1706.07162 [cs].
Rezende, Danilo Jimenez and Shakir Mohamed (2016). "Variational Inference with Normalizing Flows". In: arXiv: 1505.05770 [cs, stat].
Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). "Stochastic Backpropagation and Approximate Inference in Deep Generative Models". In: arXiv: 1401.4082 [cs, stat].
Slizovskaia, Olga, Leo Kim, Gloria Haro, and Emilia Gomez (2019). "End-to-End Sound Source Separation Conditioned On Instrument Labels". Version 2. In: arXiv: 1811.01850 [cs, eess].
Stoller, Daniel, Sebastian Ewert, and Simon Dixon (2018). "Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation" (Paris, France).

Stöter, Fabian-Robert, Antoine Liutkus, and Nobutaka Ito (2018). “The 2018 Signal Separation Evaluation Campaign”. In: arXiv: 1804.06267 [cs, eess].

Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich (2014). “Going Deeper with Convolutions”. In: arXiv: 1409.4842 [cs].

Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich (2015). “Going Deeper with Convolutions”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9.

Tabak, Esteban and Cristina V. Turner (2013). “A family of nonparametric density estimation algorithms”. In: *COMMUN. PURE & APPL. MATHS.* 66.2, pp. 145–164.

Van den Oord, Aäron, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu (2016). “WaveNet: A Generative Model for Raw Audio”. In: arXiv: 1609.03499 [cs].

Van den Oord, Aäron, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu (2016). “Conditional Image Generation with PixelCNN Decoders”. In: arXiv: 1606.05328 [cs].

Van den Oord, Aäron, Oriol Vinyals, and Koray Kavukcuoglu (2017). “Neural Discrete Representation Learning”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., pp. 6306–6315.

Welling, Max and Yee Whye Teh (2011). “Bayesian Learning via Stochastic Gradient Langevin Dynamics”. In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. ICML’11. Bellevue, Washington, USA: Omnipress, pp. 681–688 (cit. on pp. 6, 9).

Yu, Fisher and Vladlen Koltun (2016). “Multi-Scale Context Aggregation by Dilated Convolutions”. In: arXiv: 1511.07122 [cs].