

Unsupervised source separation with deep priors

Maurice Frank

August 8, 2020

Contents

Abstract

Research Question

Source separation is the task of finding a set of latent sources $\mathbf{s} = [s_1, \dots, s_k, \dots, s_N]^T \in \mathbb{R}^{N \times T}$ to an observed mix of those sources $\mathbf{m} \in \mathbb{R}^{1 \times T}$. The induced model proposes a mixing function $\mathbf{m} = f(\mathbf{s})$. The task is to find an approximate inverse model $g(\cdot)$ which retrieves \mathbf{s} :

$$\mathbf{m} = f(\mathbf{s}) \quad (1)$$

$$g(\mathbf{m}) \cong \mathbf{s} \quad (2)$$

In this learning setting *supervision* can happen in three ways: First the source signals are identified as being from class k ¹. Second the tuples (\mathbf{m}, \mathbf{s}) are supervised giving us examples of mixes and their corresponding sources. And third knowing number k of sources. The first is not a supervision signal of the separation task and therefore **is there something missing here? I guess so.**

¹ For the setting of music think of the classes being {guitar, piano, voice, ...}

1. Can we learn a source separation model $g(\cdot)$ by learning deep priors for the different source classes. **If it is a question, it should have a question mark**
2. Can we reduce this to an unsupervised setting. Unsupervised relating to the missing pairings of sources and mixes.

Background

In this section we introduce the minimally necessary background concepts our work is building upon. Our work marries two fields of research: on the one side generative, graphical models. With those one tries to model the data distribution of a \mathbf{x} observed variable by inducing dependencies of latent (unobserved) variables generating the observed variables. Modeling those dependencies, the meaningful latent variables allow us to build an intuitive generative story for the probabilistic model of the observed data. On the other hand we introduce the recent body of work on how to practically model sound signals, in particular how this compares differently to modeling images.

First we introduce the concept of deep latent-variable models, which important classes of models exist and how to train them. Second we will shortly draw up the difficulties of modeling sound and which recent solutions exist and how they differ.

Deep Latent-Variable Models

In this section we introduce the idea of latent variable and the idea of finding models of data using those latent variables. Further we discuss what practical ways are currently successful in estimating the model parameters and how they are trained.

For our process, we have observations from the data space $\mathbf{x} \in \mathcal{D}$ for which there exists an unknown data probability distribution $p^*(\mathcal{D})$. We collect a data set $\{\mathbf{x}_1 \dots \mathbf{x}_N\}$ with N samples. Further we introduce an approximate model with the density² $p_\theta(\mathcal{D})$ and model parameters θ . Learning or modeling means finding the values for θ which will give the closest approximation of the true underlying process:

$$p_\theta(\mathcal{D}) \approx p^*(\mathcal{D}) \quad (3)$$

The model p_θ has to be complex enough to be able to fit the data density while little enough parameters to be learned. Every choice for the form of the model will *induce* biases³ about what density we can model, even before we maximize a learning objective using the parameters θ .

In the following described models we assume the sampled data points \mathbf{x} to be drawn from \mathcal{D} *independent and identically distributed*⁴.

² We write density and distribution interchangeably to denote a probability function.

³ called *inductive biases*

⁴ meaning the sample of one datum does not depend on the other data points and we all draw them the same way

Therefore we can write the data log-likelihood as:

$$\log p_{\theta}(\mathcal{D}) = \sum_{x \in \mathcal{D}} \log p_{\theta}(x) \quad (4)$$

The maximum likelihood estimation of our model parameters maximizes this objective.

To form a latent-variable model we introduce a *latent variable*⁵. This latent variable can be any random variable underlying the generation of a data sample in \mathcal{D} . For example if we look at the generative story of a sound sample, a latent variable could be the sound source (the *instrument*). One can directly model the distribution of all musical notes played by all possible instruments. But we could also introduce the instrument as an underlying latent variable z , which would then lead to one distribution over this categorical variable and separate conditional densities of the sounds of each instrument. One of them then modeling the distribution of sounds made by a harp, for example.

Now the data likelihood is the marginal density of the joint latent density. With one latent variable we get:

$$p_{\theta}(x) = \int p_{\theta}(x, z) dz \quad (5)$$

Typically we introduce a factorization of the joint. The most common one and also the one from our previous example is:

$$p_{\theta}(x) = \int p_{\theta}(x|z)p(z)dz \quad (6)$$

This corresponds to the graphical model in which z is generative parent node of the observed x , see Figure 1. The density $p(z)$ is called the *prior distribution*.

If the latent is small, discrete, it might be possible to directly marginalize over it. If for example, z is a discrete random variable and the conditional $p_{\theta}(x|z)$ is a Gaussian distribution than the data model density $p_{\theta}(x)$ becomes a mixture-of-Gaussians, which we can directly estimate by maximum likelihood estimation of the data likelihood.

For more complicated models the data likelihood $p_{\theta}(x)$ as well as the model posterior $p_{\theta}(z|x)$ are intractable because of the integration over the latent z in Equation (6).

To formalize the search for an intractable posterior into a tractable optimization problem we can follow the *variational principle*⁶ which introduces an approximate posterior distribution $q_{\phi}(z|x)$, also called the *inference model*. Again the choice of the model here carries inductive biases, as such that even in asymptotic expectation we can not obtain the true posterior.

Following the derivation in [7p. 20]kingmaIntroduction2019 we introduce the inference model into the data likelihood⁸:

⁵ Latent variables are part of the directed graphical model but not observed.

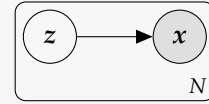


Figure 1: The graphical model with a introduced latent variable z . Observed variables are shaded.

⁶ Jordan et al., “An Introduction to Variational Methods for Graphical Models”, 1999.

⁷ [.

⁸ The first step is valid as q_{θ} is a valid density function and thus integrates to one.

$$\log p_{\theta}(x) = \mathbb{E}_{q_{\theta}(z|x)} [\log p_{\theta}(x)] \quad (7)$$

$$= \mathbb{E}_{q_{\theta}(z|x)} \left[\log \frac{p_{\theta}(x, z)}{p_{\theta}(z|x)} \right] \quad (8)$$

$$= \mathbb{E}_{q_{\theta}(z|x)} \left[\log \frac{p_{\theta}(x, z)}{q_{\phi}(z|x)} \frac{q_{\phi}(z|x)}{p_{\theta}(z|x)} \right] \quad (9)$$

$$= \mathbb{E}_{q_{\theta}(z|x)} \left[\log \frac{p_{\theta}(x, z)}{q_{\phi}(z|x)} \right] + \mathbb{E}_{q_{\theta}(z|x)} \left[\log \frac{q_{\phi}(z|x)}{p_{\theta}(z|x)} \right] \quad (10)$$

$$= \mathbb{E}_{q_{\theta}(z|x)} \left[\log \frac{p_{\theta}(x, z)}{q_{\phi}(z|x)} \right] + \mathbb{D}_{\text{KL}}[q_{\phi}(z|x) \| p_{\theta}(z|x)] \quad (11)$$

Note that we separated the likelihood into two parts. The second part is the (positive) Kullback-Leibler divergence of the approximate posterior from the true intractable posterior. This unknown divergence states the ‘correctness’ of our approximation⁹. **It could be clearer that the ELBO is a lower bound for the likelihood.**

The first term is the *variational free energy* or *evidence lower bound* (ELBO):

$$\text{ELBO}_{\theta, \phi}(x) = \mathbb{E}_{q_{\theta}(z|x)} \left[\log \frac{p_{\theta}(x, z)}{q_{\phi}(z|x)} \right] \quad (12)$$

We can introduce the same factorization as in Equation (6):

$$\text{ELBO}_{\theta, \phi}(x) = \mathbb{E}_{q_{\theta}(z|x)} \left[\log \frac{p_{\theta}(x|z)p(z)}{q_{\phi}(z|x)} \right] \quad (13)$$

$$= \mathbb{E}_{q_{\theta}(z|x)} \left[\log \frac{p(z)}{q_{\phi}(z|x)} \right] + \mathbb{E}_{q_{\theta}(z|x)} [\log p_{\theta}(x|z)] \quad (14)$$

$$= -\mathbb{D}_{\text{KL}}[q_{\phi}(z|x) \| p(z)] + \mathbb{E}_{q_{\theta}(z|x)} [\log p_{\theta}(x|z)] \quad (15)$$

Under this factorization, we separated the lower bound into two parts. First, the divergence of the approximate posterior from the latent prior distribution and second the data posterior likelihood from the latent¹⁰.

The optimization of the $\text{ELBO}_{\theta, \phi}$ allows us to jointly optimize the parameter sets θ and ϕ . The gradient with respect to θ can be estimated with an unbiased Monte Carlo estimate using data samples¹¹. Though we can *not* do the same for the variational parameters ϕ , as the expectation of the ELBO is over the approximate posterior which depends on ϕ . By a change of variable of the latent variable we can make this gradient tractable, the so called *reparameterization trick*¹². We express the $z \sim q_{\theta}$ as a random sample from a unparametrized source of entropy ϵ and a parametrized transformation:

$$z = f_{\eta}(\epsilon) \quad (16)$$

For example for a Gaussian distribution we can express $z \sim \mathcal{N}(\mu, \sigma)$ as $z = \mu + \sigma \cdot \epsilon$ with $\epsilon \sim \mathcal{N}(0, 1)$ and $\eta = \{\mu, \sigma\}$.

⁹ More specifically the divergence marries two errors of our approximate model. First, it gives the error of our posterior estimation from the true posterior, by definition of divergence. Second, it specifies the error of our complete model likelihood from the marginal likelihood. This is called the *tightness* of the bound.

¹⁰ This will later be the reconstruction error. How well can we return to the data density from latent space?

¹¹ $\nabla_{\theta} \text{ELBO}_{\theta, \phi} \cong \nabla_{\theta} \log p_{\theta}(x, z)$

¹² Kingma and Welling, “Auto-Encoding Variational Bayes”, 2014.

The VAE framework

The variational autoencoder (VAE) ^{12,13} is the previously introduced ideas together into a trainable model.

more

```

1: while Training is not converged do
2:    $X \sim \mathcal{D}$  ▷ Sample random mini-batch
3:    $\eta \leftarrow \text{EncoderNN}(X)$ 
4:    $\epsilon \sim p(\epsilon)$ 
5:    $z \leftarrow f_\eta(\epsilon)$ 
6:    $X' \leftarrow \text{DecoderNN}(z)$ 
7: end while

```

The β -VAE¹⁴ extends the VAE objective with an β hyperparameter in front of the KL divergence. The value β gives a constraint on the latent space, controlling the capacity of it. Adapting β gives a trade-off between the reconstruction quality of the autoencoder and the simplicity of the latent representations¹⁴. Using such a constraint is similar to the use of the information bottleneck¹⁵.

Flow based models

Another class of common deep latent models is based on *normalizing flows*¹⁶. A normalizing flow is a function $f(x)$ that maps the input density to a fixed, prescribed density $p(\epsilon) = p(f(x))$, in that normalizing the density¹⁷. They use a flow for the approximate posterior $q_\phi(z|x)$. Again this is commonly set to be a factorized Gaussian distribution.

For a finite normalizing flow, we consider a chain of invertible, smooth mappings.

NICE¹⁸ - volume preserving transformations - coupling layer - triangular shape

Normalizing Flow¹⁹

RealNVP²⁰ builds on top of NICE creating a more general, non-volume preserving, normalizing flow.

$$y_{1:d} = x_{1:d} \quad (17)$$

$$y_{d+1:D} = x_{d+1:D} \odot \exp(s(x_{1:d})) + t(x_{1:d}) \quad (18)$$

$$\frac{\partial y}{\partial x^T} = \begin{bmatrix} \mathbb{1}_d & 0 \\ \frac{\partial y_{d+1:D}}{\partial x_{1:d}^T} & \text{diag}(\exp(s(x_{1:d}))) \end{bmatrix} \quad (19)$$

Glow²¹ extended the RealNVP by introducing invertible 1×1 -convolutions. Instead of having fixed masks and permutations for the computations of the affine parameters in the coupling layer, Glow learns a rotation matrix which mixes the channels. After mixing the input can always be split into the same two parts for the affine transformation. Further, the authors showed that training can

¹³ Rezende, Mohamed, and Wierstra, "Stochastic Backpropagation and Approximate Inference in Deep Generative Models", 2014.

Algorithm 1: Training's procedure for a variational autoencoder

¹⁴ Higgins et al., "Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework", 2016.

¹⁵ Burgess et al., "Understanding Disentangling in Beta-VAE", 2018.

¹⁶ Tabak and Turner, "A family of nonparametric density estimation algorithms", 2013.

¹⁷ The extreme of this idea is, of course, an infinitesimal, continuous-time flow with a velocity field.

¹⁸ Dinh, Krueger, et al., "NICE: Non-Linear Independent Components Estimation", 2015.

¹⁹ Rezende and Mohamed, "Variational Inference with Normalizing Flows", 2016.

²⁰ Dinh, Sohl-Dickstein, et al., "Density Estimation Using Real NVP", 2017.

²¹ Kingma and Dhariwal, "Glow: Generative Flow with Invertible 1x1 Convolutions", 2018.

be helped by initializing the last layer of each affine parameter network with zeros. This ensures that at the beginning without weight update each coupling layer behaves as an identity.

Langevin dynamics

In stochastic optimization we update the parameters θ with the gradient of the posterior using a subset of samples from our data set.

$$\Delta\theta_t = \frac{\epsilon_t}{2} \{(\nabla \log p(\theta_t)) + \frac{N}{n} \sum_{i=1}^n \nabla \log p(x_{ti}|\theta_t)\} \quad (20)$$

This will lead to the maximum a posteriori (MAP) estimates of the parameters θ . By taking the gradient of the posterior only under a subset of the data will give us a noisy estimate of the real gradient, but is computationally cheaper. The idea is, that the noise of the noisy gradients smoothes out while sampling different subsets. When using a decaying step size ϵ_t the stochastic optimization is guaranteed to converge to a local maximum.

Now the Langevin dynamics²² provides a Markov chain Monte Carlo (MCMC) sampling technique for estimating the parameters of a distribution without overfitting on the data(??).

²² Neal, "MCMC Using Hamiltonian Dynamics", 2012.

$$\Delta\theta_t = \frac{\epsilon}{2} \{(\nabla \log p(\theta_t) + \frac{N}{n} \sum_{i=1}^N \nabla \log p(x_i|\theta_t))\} + \eta_t \quad (21)$$

$$\eta_t \sim \mathcal{N}(0, \epsilon) \quad (22)$$

$$\Delta\theta_t = \frac{\epsilon}{2} \{(\nabla \log p(\theta_t) + \sum_{i=1}^N \nabla \log p(x_i|\theta_t))\} + \eta_t \quad (23)$$

$$\eta_t \sim \mathcal{N}(0, \epsilon) \quad (24)$$

Modeling audio

In the physical world a sound signal is a change of the density of the carrier air over time. As such it is a one-dimensional temporal signal²³. Humans sound perception is *stereo* as we have two ears and sense the changes in pressure at two different points in space simultaneously.

Representations of audio

The simplest form of digitally representing a sound signal is recording the amplitude at fixed intervals at one or multiple points in space with a microphone. This method is called Pulse-Code modulation (PCM). It introduces two parameters which will bias the result of the record. First we have to pick a temporal frequency with which the pressure samples are taken, the so-called sample rate. The

²³ Sound waves in gases are purely compressive, therefore they cannot be polarized which would introduce a higher complexity. In solids a sound signal can have polarization, think of an earthquake with shear and pressure components.

Nyquist-Shannon theorem²⁴ tells us that if the highest frequency in the true signal is B Hz then with a sample rate of $2 \cdot B$ Hz we will capture the complete signal with all frequencies. At a lower sample rate we might introduce aliasing effects. Second we have to represent each amplitude sample as a digital value. Most commonly the sample is encoded into a 8 bit or 16 bit integer. The microphone has a range of air pressure which w.l.o.g. we set to $[-1, 1]$, where a value of 0 is no signal, 1 shows maximum compression and -1 maximum decompression of the carrier air. The simplest form of encoding is Linear PCM. In Linear PCM amplitudes are quantized on an evenly spaced grid over the possible value range. More advanced quantization include μ -law encoding²⁵ in which the quantization intervals are varied with the amplitude²⁶. As the human perception of loudness is logarithmic this encoding also quantizes the amplitude values on a logarithmic response curve. Simply speaking the lower amplitudes will be quantized on a finer grid than higher amplitudes. This makes it possible to achieve a higher signal-to-noise ratio at smaller encoding sizes for sound signals.

As an example, if encoding a sound signal with 8 kHz 16 bit Linear PCM we sample an amplitude value every $\frac{1}{8000}$ s and quantize those into $2^{16} = 65536$ bins. A sample with 100 values is only 12.5 ms long!

A standard full piano with 88 keys tuned to 440 Hz at the A_4 ²⁷ has a frequency range of [27.5 Hz, 4186 Hz] from the A_0 to the C_8 , respectively. Therefore just for the recording of a standard piano we need a sampling rate of more than 8 kHz. Beyond the piano many instruments exhibit even larger frequency content which leads to the conclusion that not just for recording but also for our goal of modeling musical sounds a reasonably high sampling rate is crucial.

Looking at the temporal scales of musical information, on the small end we have the modeling of pitch as discussed before which happens at scales of $[10^{-1} \text{ ms}, 10^2 \text{ ms}]$ over single notes happening at scales of $[10^2 \text{ ms}, 10^3 \text{ ms}]$ to the structure of melodies and full songs happening at $[10^4 \text{ ms}, 10^6 \text{ ms}]$. The modeling of music happens over 7 magnitudes of time! In ?? we will present common solutions to this difficulty.

Spectrograms

Commonly used alternative representations of signals in sound processing are spectrograms. Spectrograms transport the signal as it is in time-domain into the frequency domain.

Time-domain vs spectrogram, what is a spectrogram, missing phase information, complex spectrograms

Modeling raw audio

Deep learning models as used for image applications are unsuitable for raw audio signals (signals in *time-domain*). Digital audio is

²⁴ Kotelnikov, "On the Carrying Capacity of the Ether and Wire in Telecommunications", 1933.

²⁵ *Pulse Code Modulation (PCM) of Voice Frequencies*, 1972.

²⁶ The actual formula for calculating the μ -law quantization is:

$$\mu\text{law}(x) = \text{sgn}(x) \cdot \frac{\ln(1 + \mu|x|)}{\ln(1 + \mu)}$$

²⁷ ISO/TC 43 Acoustics, *ISO 16 Acoustics Standard Tuning Frequency*, 1975.

sampled at high sample rates, commonly 16kHz up to 44kHz. The features of interest lie at scales of strongly different magnitudes. Recognizing the local-structure of a signal, like frequency and timbre, might require features at short intervals (\approx tens of milliseconds) but modeling of speech or music features happens at the scale of seconds to minutes. As such a generative model for this domain has to model at these different scales.

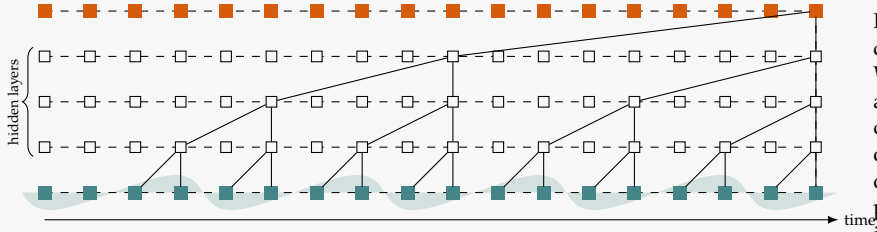


Figure 2: An example of how dilated convolutions are used in the WaveNet. We see three hidden layers with each a kernel size of two. By using the dilations the prediction of the new output element has a receptive field of 18. This convolution is *causal* as the prediction depends only on previous input values. Causality is enforced through asymmetric padding.

The **WaveNet**²⁸ introduced an autoregressive generative model for raw audio. It is build upon the similar PixelCNN²⁹ but adapted for the audio domain. The WaveNet accomplishes this by using dilated causal convolutions. Using a stack of dilated convolutions increases the receptive field of the deep features without increasing the computational complexity, see Figure 2. Further, the convolutions are gated and the output is constructed from skip connections. For the structure of a hidden layer refer to Figure 3. A gated feature, as known from the LSTM³⁰, computes two outputs: one put through an sigmoid $\sigma(\cdot)$ activation and one through an $\tanh(\cdot)$ activation. The idea being that the sigmoid (with an output range of $[0, 1]$) regulates the amount of information, thereby gating it, while the \tanh (with a range of $[-1, 1]$) gives the magnitude of the feature. The output of the WaveNet is the sum of outflowing skip connections added after each (gated) hidden convolution. This helps fusing information from multiple time-scales (*low-level* to *high-level*) and makes training easier³¹. The original authors tested the model on multiple audio generation tasks. They used a μ -law encoding³² which discretizes the range $[-1, 1]$ to allow a set of μ targets and an multi-class cross-entropy training objective. While being quite unnatural this is done to avoid making any assumptions about the target distribution. Sound generation with a WaveNet is slow as the autoregressiveness requires the generation value by value. The original WaveNet setting is generating waveforms, by giving the previously generated values as the input and conditioning the process on target classes, $p(x_t|x_{1:t}, c_t)$. Therefore the generation has to happen value-by-value (autoregressive). This can be alleviated by keeping intermediate hidden activations cached³³. The WaveNet can be conditioned by adding the weighted conditionals in the gate and feature activations of the gated convolutions³⁴.

add PixelCNN++?

The first generative sounds model to employ a WaveNet is **NSynth**³⁵. They construct a VAE where encoder and decoder are both WaveNet-

²⁸ Van den Oord, Dieleman, et al., “WaveNet: A Generative Model for Raw Audio”, 2016.

²⁹ Van den Oord, Kalchbrenner, et al., “Conditional Image Generation with PixelCNN Decoders”, 2016,

³⁰ Hochreiter and Schmidhuber, “Long Short-Term Memory”, 1997.

³¹ Szegedy et al., “Going Deeper with Convolutions”, 2014.

³² Pulse Code Modulation (PCM) of Voice Frequencies, 1972.

³³ Paine et al., “Fast Wavenet Generation Algorithm”, 2016.

³⁴ Van den Oord, Kalchbrenner, et al., “Conditional Image Generation with PixelCNN Decoders”, 2016.

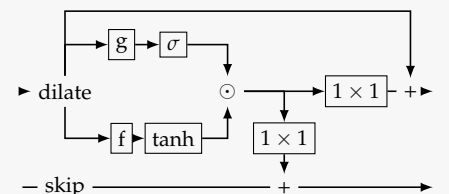


Figure 3: Hidden layer as in the

like. The so-called *non-causal temporal encoder* uses a stack of dilated residual non-causal convolutions. The convolutions are not gated and no skip-connections are used. The decoder is a WaveNet taking the original input chunk as an input and predicts the next value of the sound sample, while being conditioned on the latent variable. The authors use this model to learn latent temporal codes from a new large set of notes played by natural and synthesized instruments. The latent of the VAE is conditioned on the pitch of these notes. While the model is difficult to train, they show great improvement of the WaveNet-based VAE compared to a spectral-based VAE.

Chorowski et al.³⁶ presents another WaveNet-based VAE. They are learning speech representations, unsupervised. The encoder is a residual convnet and takes the MFCC of the signal as its input. As the bottleneck they found a VQ codebook³⁷ to be most successful. The decoder is an autoregressive WaveNet conditioned on the latent features. Note again that this model infers the latent features from mel-cepstra but reconstructs the signal with the WaveNet in time-domain.

In Prenger et al.³⁸

FloWaveNet Kim et al.³⁹

³⁶ Chorowski et al., “Unsupervised Speech Representation Learning Using WaveNet Autoencoders”, 2019.

³⁷ Van den Oord, Vinyals, et al., “Neural Discrete Representation Learning”, 2017.

³⁸ Prenger et al., “WaveGlow: A Flow-Based Generative Network for Speech Synthesis”, 2018.

³⁹ Kim et al., “FloWaveNet : A Generative Flow for Raw Audio”, 2019.

Related work

In this section we give an overview of recent approaches to (sound) source separation which are related to our own approach either in the chosen model choices or implicit ideas. As the body of research into this task, especially in practical application, is vast and diverse, this overview can only be a small insight into tangentially related and recent research work.

Source separation

All here presented model maximize $p(s|m)$ for one source (e.g. extracting only the singing voice out of a mix) or $p(s_1, \dots, s_N|m)$ for multiple sources. Note that in the second case the conditional likelihood is not factorized, meaning we build a shared model for all sources.

WaveNet based

Rethage et al.⁴⁰ use a WaveNet for speech denoising. Speech denoising is a special case of source separation as the observed mix is separated into true signal and noise. The authors made the WaveNet non-causal by making the padding symmetrical. Further they used L_1 -loss, instead of the multi-class μ -law objective. They show that for their case the real valued predictions behave better.

Lluís et al.⁴¹ adapted a time-domain WaveNet for musical source separation. The non-causal ⁴² WaveNet directly outputs the separated sources and is trained fully supervised. They show this simple setup performing well against spectrogram based baselines. Also, the experiments show a higher performance with a network that is deeper but less wide ⁴³.

U-Net based

Jansson et al.⁴⁴ were the first to use an U-Net architecture (see Figure 4) for musical source separation. They used a convolutional U-Net on spectrograms of musical data to extract the singing voice. Input to the network is the spectral magnitude and the output prediction is an equally-sized masked. The voice signal reconstruction is then done by multiplying the input magnitude with the predicted mask and using the original phase information from the mix, unal-

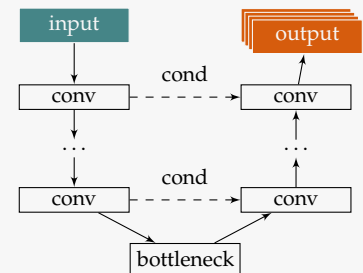


Figure 4: High level idea of a U-Net: the input gets projected into an (informational) bottleneck through some form of convolutional architecture. From this bottleneck the predictions are from an mirrored array of upsampling convolutional layer (either tuples of upsampling and convolutions or through dilated convolutions). The intermediate filter activations of the encoding are used as conditional inputs at the respective output scaling.

⁴⁰ Rethage et al., “A Wavenet for Speech Denoising”, 2018.

⁴¹ Lluís et al., “End-to-End Music Source Separation: Is It Possible in the Waveform Domain?”, 2019.

⁴² In this setting the WaveNet can be non-causal because the prediction happens from the given mix and is not autoregressive.

⁴³ Deepness of a network refers to the number of hidden layers. Wideness refers to the number of kernels/features of each of these hidden layers. Making the WaveNet deeper significantly increases its receptive field.

⁴⁴ Jansson et al., “Singing Voice Separation with Deep U-Net Convolutional Networks”, 2017.

tered. The training objective is the L_1 loss between the masked input spectrogram and the target signal.

The Wave-U-Net⁴⁵ brings this idea into the time-domain. The downstreaming and upstreaming layers are replaced with WaveNet style 1D convolutional layers. Further here the model is predicting multiple source signals.

Slizovskaia et al.⁴⁶ extends the Wave-U-Net by conditioning the filter activations at the bottleneck with the instrument class label. The additional information is improving the separation quality.

Cohen-Hadria et al.⁴⁷ show improvements of training a Wave-U-Net when using additional data augmentation. They apply a hand-crafted set of auditory transformations (pitch-shifting, time-stretching, transforming the spectral envelope of the singing voice) to the training set of the musdb18 dataset.

Kaspersen⁴⁸ adds an bidirectional LSTM at the bottleneck of the Wave-U-Net. The BiLSTM is capable of select and keep information at the bottleneck over time and combine this memory with the new encoded information. This makes it possible for the network to keep information about the source signal at scales larger than the actual receptive field of the U-Net. The research shows an significant improvement over previous U-Net based approaches.

⁴⁹

Demucs⁵⁰ also introduces a BiLSTM at the bottleneck⁵¹. Additionally to the ideas from the HydraNet Demucs also adds data augmentation and makes additional internal changes to the network architecture. They simplify the layer blocks by using the simpler Gated Linear Units⁵² and while the Wave-U-Net is using upsampling followed by a convolutional layer in the decoder, here the others directly use transposed convolutions to achieve the upsampling. Demucs is the first time-domain based end-to-end model that is achieving similar or better results compared to spectrogram based models.

Auto-encoder based

Grais et al.⁵³ present the first work using an auto-encoder for multi-channel sound source separation. Their auto-encoder takes raw audio as the input is using multi-scale convolutional layers (transposed in the decoder) in the encoder and decoder, combining the activations of the different scales before the activation. (Similar to Inception networks⁵⁴).

⁴⁵ Stoller et al., "Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation", 2018.

⁴⁶ Slizovskaia et al., "End-to-End Sound Source Separation Conditioned On Instrument Labels", 2019.

⁴⁷ Cohen-Hadria et al., "Improving Singing Voice Separation Using Deep U-Net and Wave-U-Net with Data Augmentation", 2019.

⁴⁸ Kaspersen, "HydraNet: A Network For Singing Voice Separation", 2019.

⁴⁹ Narayanaswamy et al., "Audio Source Separation via Multi-Scale Learning with Dilated Dense U-Nets", 2019.

⁵⁰ Défossez, Usunier, et al., "Demucs: Deep Extractor for Music Sources with Extra Unlabeled Data Remixed", 2019.

⁵¹ Défossez, Zeghidour, et al., "SING: Symbol-to-Instrument Neural Generator", 2018.

⁵² Dauphin et al., "Language Modeling with Gated Convolutional Networks", 2017.

⁵³ Grais et al., "Raw Multi-Channel Audio Source Separation Using Multi-Resolution Convolutional Auto-Encoders", 2018.

⁵⁴ Szegedy et al., "Going Deeper with Convolutions", 2014.

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.

We propose the graphical model as shown in Figure 5 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 *Research Question* is to retrieve the sources $\{s_1, \dots, s_N\}$ from a given mix m . Our model is trained without using sample tuples

$$(m, \{s_1, \dots, s_N\}) : f(\{s_1, \dots, s_N\}) = m$$

which would show the relation between separate sources and mixes. The general idea is visualized in Figure 6.

Looking back at the graphical model in Figure 5, it implies the following factorization:

$$p(m) = \int^N p(s_1, \dots, s_N, m) d^N s \quad (25)$$

$$= \int^N p(m|s_1, \dots, s_N) \cdot p(s_1, \dots, s_N) d^N s \quad (26)$$

$$= \int^N p(m|s_1, \dots, s_N) \cdot p(s_1) \cdots p(s_N) d^N s \quad (27)$$

While the conditional $p(m|s_1, \dots, s_N)$ is not even probabilistic, as the mix is generated deterministically with the mean of the sources, the model posterior $p(s_1, \dots, s_N|m)$ 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 $p(m)$ 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.

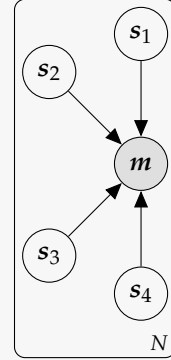


Figure 5: The used graphical model for the source separation task. We have the latent source channel variables s_k . Exemplary here, as in our data, we have four sources. The mix m is observed.

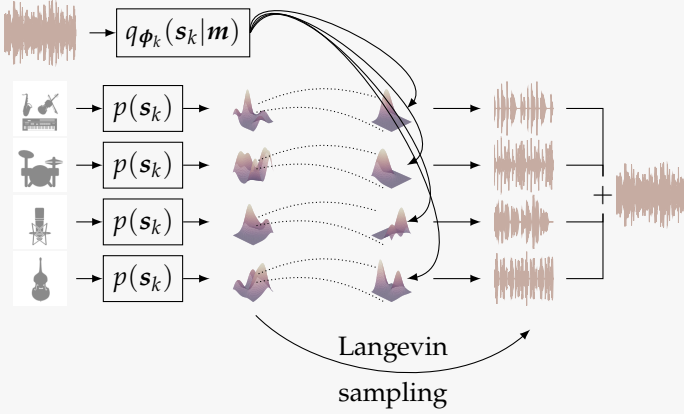


Figure 6: The method visualized, **not finished yet**

Already with the graphical model we introduce a fairly big assumption, namely that we can model the source distributions independently from each other when modeling the joint:

$$p(\mathbf{m}, s_1, \dots, s_N) \equiv p(\mathbf{m}|s_1, \dots, s_N) \cdot \prod_k^N p(s_k) \quad (28)$$

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 densities⁵⁵. Nevertheless, this assumption is crucial to be able to model the prior instrumental models without needing the tuples (s_1, \dots, s_N) of co-occurring stems.

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 $p(s_k)$ for each source k . Thereout we need to choose a model which gives the parametrized approximated prior $p_{\theta}(s_k)$ with the parameters θ which we optimize with the samples from \mathcal{D}_k .

Second, for each source, there exists a posterior given a mix $p(s_k|m)$ from which we want to draw samples. Here two approaches are possible. Either we can propose an approximate posterior $q_{\phi_k}(s_k|m)$, 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)⁵⁶ without explicitly approximating the posterior distribution (sampling setting). Both optimization, either training or sampling, happen under the mixing constraint:

$$\sum_{k=1}^N s_k = \mathbf{m} \quad (29)$$

When using the VAE setting we then can sample from each posterior to retrieve a set of (approximately) correct source tracks. In the

⁵⁵ 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.

⁵⁶ Welling and Teh, “Bayesian Learning via Stochastic Gradient Langevin Dynamics”, 2011.

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 $p_{\theta}(s_k)$ of the *prior* distribution
2. a parametrized approximate posterior distribution $q_{\phi_k}(s_k|\mathbf{m})$ with parameters η
3. a parametrized *encoder* which gives the parameters for the posterior distribution $\text{Encoder}_{\phi}(\mathbf{m}) = \eta$
4. a *decoder* which returns the input \mathbf{m} from the latents $\text{Decoder}(\{s_1, \dots, s_N\}) = \mathbf{m}$

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 $p(s_k)$. 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 *Flow based models*. The invertibility of the flow gives us the ability to sample from the priors, which is necessary for the sampling-based approach. For the variational-based approach sampling is not necessary, nevertheless we utilize the same generative priors for both experiments. 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, FloWaveNet⁵⁷ and WaveGlow⁵⁸, respectively.

Remove Glow if no experiment will use it

⁵⁷ Kim et al., “FloWaveNet : A Generative Flow for Raw Audio”, 2019.

⁵⁸ Prenger et al., “WaveGlow: A Flow-Based Generative Network for Speech Synthesis”, 2018.

For the case of the time-domain representation, the data has only one channel (mono-audio). Therefore a Glow-like shuffling 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 *Representations of audio* 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 7. We construct normalizing flow models, following the description in Kim et al.⁵⁹.

The flow network, at the highest level, consists of n_b context blocks, each of whom contain one squeeze operation and n_f flows. The squeeze operation halves the length of the signal by doubling the number of channels. By doing so, we effectively double the receptive field of the WaveNet operators in the affine couplings. One flow contains one affine coupling layer after which the masking for the coupling (*even/odd*) is inverted. Before the coupling layer, we apply activation normalization⁶⁰. The affine coupling is applying an affine transformation on the *odd* set of the input, with scale s and translation t coming from the *even* input put through a *non-causal* WaveNet.

The *even/odd* masking for the coupling layer is simply achieved by splitting the channel dimension into two equal parts. Flipping the masking exchanges the top half of channels with the bottom half. No channel transformation of the channels is learned as in Glow. Further observe that through the repeated squeeze operation the

⁵⁹ Kim et al., “FloWaveNet : A Generative Flow for Raw Audio”, 2019.

⁶⁰ Kingma and Dhariwal, “Glow: Generative Flow with Invertible 1x1 Convolutions”, 2018.

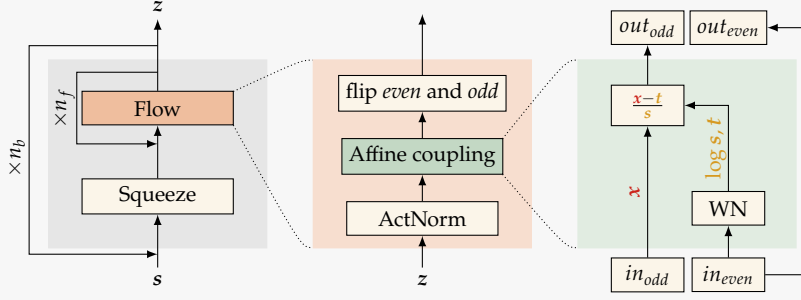


Figure 7: The building blocks for the prior model. The model consists of n_b blocks (left). Each block consists of n_f flows (middle). In each flow we apply activation normalization, followed by the affine coupling (right), after which the binary mask for the even/odd mapping is inverted. The affine coupling layer uses a WaveNet with the *even* set as the input to output scaling $\log s$ and translation t which the *odd* set is transformed.

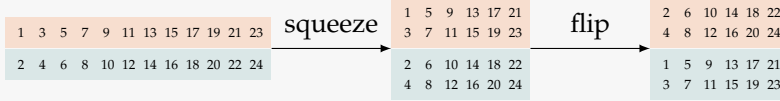


Figure 8: Flip and squeeze operation for the coupling layers. The squeeze operation doubles the channels and halves the length while the flip operation flips the split of channel dimension into *odd* and *even*.

masking implicitly includes a checkerboard pattern on the temporal dimension, similar to the explicit pattern used in RealNVP.

As explained in ?? the invertible flow $f(\cdot)$ network directly maps the data space into a prior distribution $p(z)$. Through the invertibility we can directly optimize the log probability with the change of variable formula:

$$\log p(x) = \log p(f(x)) + \log \det \left(\frac{\partial f(x)}{\partial x} \right) \quad (30)$$

Our network consists of n_b blocks with each n_f flows each with one activation normalization layer f_{AN} and an affine coupling layer f_{AC} . Therefore the log-determinant becomes:

$$\log \det \left(\frac{\partial f(x)}{\partial x} \right) = \sum_i^{n_b \cdot n_f} \log \det \frac{\partial f_{AN}(x)}{\partial x} + \log \det \frac{\partial f_{AC}(x)}{\partial x} \quad (31)$$

The activation normalization layer f_{AN} learns an affine transformation per channel i :

$$f_{AN}^i(x_i) = s_i \cdot x_i + t_i \quad (32)$$

The log-determinant is easily computed with⁶⁰:

$$\log \det \frac{\partial f_{AN}(x)}{\partial x} = T \cdot \sum_i^C \log |s_i| \quad (33)$$

where C is the number of channels and T is the length of the signal.

For the affine coupling layer, the log-determinant is made tractable by only transforming half the elements in each application. The coupling transforms the *odd* part as follows:

$$\log s, t = \text{WN}(in_{odd}) \quad (34)$$

$$out_{odd} = f_{AC}(in_{odd}) = \frac{in_{odd} - t}{s} \quad (35)$$

The inverse is:
 $f_{AN}^{-1}(y_i) = \frac{y_i - t_i}{s_i}$

The inverse is:

$$f_{AC}^{-1}(out_{odd}) = out_{odd} \cdot s + t$$

where $WN(\cdot)$ is a WaveNet. The Jacobian of the coupling function is a lower triangular matrix⁶¹ because of which the log-determinant becomes the product of the diagonal elements:

$$\log \det \frac{\partial f_{AC}(x)}{\partial x} = -\sum s \quad (36)$$

The purpose of the invertible flow is it to project the complex data distribution into a tractable defined prior distribution. In most cases, as in this, the prior $p(z)$ is set to be a factorized Gaussian distribution with zero mean and variance of one. As such the likelihood of $f(x) \sim p(z)$ can be evaluated with:

$$\log p(f(x)) = -\frac{1}{2}(\log(2 \cdot \pi)) + f(x)^2 \quad (37)$$

To train the model we maximize the likelihood in Equation (30) which we is the sum of the parts in Equations (33), (36) and (37). Optimization is done in mini-batches with the Adam optimizer⁶² and a decaying learning rate scheme.

It is noted that the likelihood under the prior is computed “pixel”-wise⁶³. While for the optimization the actual loss on which the optimizers performs update steps is the mean likelihood, we can use the trained prior flow to assign a likelihood map to an input signal where each log-likelihood value is depending on this values receptive field.

WaveNet

In the affine coupling layer we utilize a *non-causal* WaveNet for generating the scale $\log s$ and translation t values. The WaveNet is non-causal because the filters are centred, as such the receptive field of an output value is symmetrically in the past and future input space. The kernel size is fixed to 3. The WaveNet is using the original gated convolution scheme described in *Background*. Appended to the last skip connection output are two additional one-dimensional convolutional layers, each preceded by a ReLU. The last convolutional layer is initialized to zero. By that we initialize the affine transformation of the coupling layer to the identity at the beginning of training, helping to stabilize optimization. The last layer outputs both scaling and translation factors. Learning those factors with weight sharing does not change the formulation of the coupling layer, as the log-determinant does not depend on the network. Learning them in the same network might increase efficiency, as well as, increasing implementation simplicity. The WaveNet completely consists of convolutional operations. As such it can be applied to one-dimensional signals of variable length.

Sampling from the posteriors

After having estimated the N prior distributions the first possibility for retrieving sample estimates s_k is sampling from the posterior

⁶¹ Dinh, Sohl-Dickstein, et al., “Density Estimation Using Real NVP”, 2017.

⁶² Kingma and Ba, “Adam: A Method for Stochastic Optimization”, 2017.

⁶³ time-points in our sound case

using Langevin dynamics⁶⁴. With Stochastic Gradient Langevin Dynamics (SGLD) we can sample from $s_i \sim p(\cdot|\mathbf{m})$ without computing, explicitly, the posterior $p(s_i|\mathbf{m})$. For a overview of SGLD see *Langevin dynamics*.

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

$$s_k^{(t+1)} = s_k^{(t+1)} + \eta \cdot \nabla_{s_k} \log p(s_k^{(t)}|\mathbf{m}) + 2\sqrt{\eta}\epsilon_t \quad (38)$$

$$= s_k^{(t+1)} + \eta \cdot \nabla_{s_k} [\log p(\mathbf{m}|s_k) + \log p(s_k) - \log p(\mathbf{m})] + 2\sqrt{\eta}\epsilon_t \quad (39)$$

$$= s_k^{(t+1)} + \eta \cdot \nabla_{s_k} [\log p(\mathbf{m}|s_k) + \log p(s_k)] + 2\sqrt{\eta}\epsilon_t \quad (40)$$

$$= s_k^{(t+1)} + \eta \cdot \left[\|\mathbf{m} - \frac{1}{N} \sum_j s_j^{(t)}\| + \nabla_{s_k} \log p(s_k) \right] + 2\sqrt{\eta}\epsilon_t \quad (41)$$

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

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. See Algorithm 2 for the algorithm.

```

1: for  $t = 1 \dots T$  do
2:   for  $k = 1 \dots N$  do
3:      $\epsilon_t \sim \mathcal{N}(0, \mathbb{I})$ 
4:      $\Delta s_k^t \leftarrow s^t + \eta \cdot \nabla \log p(s^t) + 2\sqrt{\eta}\epsilon_t$ 
5:   end for
6:   for  $k = 1 \dots N$  do
7:      $s_k^{t+1} \leftarrow \Delta s_k^t - \frac{\eta}{\sigma^2} \cdot [\mathbf{m} - \frac{1}{N} \sum_i s_i^t]$ 
8:   end for
9: end for

```

The idea of using SGLD in combination with deep parametrized priors for source separation was, concurrently to this work, introduced in Jayaram and Thickstun⁶⁵. The authors empirically show that

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

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.

Will add more discussion of their results

⁶⁴ Welling and Teh, “Bayesian Learning via Stochastic Gradient Langevin Dynamics”, 2011.

$\log p(\mathbf{m})$ is independent from s_k therefore no gradient.

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 .

⁶⁵ Jayaram and Thickstun, “Source Separation with Deep Generative Priors”, 2020.

Modeling the posteriors

Instead of formulating a sampling regime for approaching s_k we can also use the prior models to variationally train a model to estimate the parameters of an approximate posterior $q_{\phi_k}(s_k|\mathbf{m})$. While estimating the true posterior $p(s_k|\mathbf{m})$ 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 $q_{\phi_k}(s_k|\mathbf{m})$ for each source channel. Next, we express the mix density as the expectation over those posteriors:

$$\log p(\mathbf{m}) = \mathbb{E}_{q_{\phi_k}(s_k|\mathbf{m})}^N [\log p(\mathbf{m})] \quad (42)$$

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

$$\mathbb{E}_{q_{\phi_k}(s_k|\mathbf{m})}^N [\log p(\mathbf{m})] = \mathbb{E}_{q_{\phi_k}(s_k|\mathbf{m})}^N \left[\log \frac{p(\mathbf{m}, s_1, \dots, s_N)}{p(s_1, \dots, s_N|\mathbf{m})} \right] \quad (43)$$

$$= \mathbb{E}_{q_{\phi_k}(s_k|\mathbf{m})}^N \left[\log \frac{p(\mathbf{m}|s_1, \dots, s_N) \cdot \prod_k^N p(s_k)}{\prod_k^N q_{\phi_k}(s_k|\mathbf{m})} + \log \frac{\prod_k^N q_{\phi_k}(s_k|\mathbf{m})}{p(s_1, \dots, s_N|\mathbf{m})} \right] \quad (44)$$

Note that again we use the assumption in Equation (28) 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 (42) over those is still correct. The error arising from the independence assumption is captured in the tightness of the ELBO.

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

$$\mathbb{E}_{q_{\phi_k}(s_k|\mathbf{m})}^N [\log p(\mathbf{m})] \geq \sum_k^N \mathbb{E}_{q_{\phi_k}(s_k|\mathbf{m})} \left[\log \frac{p(s_k)}{q_{\phi_k}(s_k|\mathbf{m})} \right] + \mathbb{E}_{q_{\phi_k}(s_k|\mathbf{m})} [p(\mathbf{m}|s_1, \dots, s_N)] \quad (45)$$

The lower bound, as in the normal VAE, will be our optimization target for training the inference models that give the variational parameters ϕ_k . We can also formulate the objective for just one source, as the Encoders are trained independently⁶⁶. Thus we come to:

⁶⁶ While the optimization of the Encoders is independent, the training is, of course, concurrent as the mixing condition depends on all N source samples.

$$\mathbb{E}_{q_{\phi_k}(s_k|\mathbf{m})} [\log p(\mathbf{m})] \geq \mathbb{E}_{q_{\phi_k}(s_k|\mathbf{m})} \left[\log \frac{p(s_k)}{q_{\phi_k}(s_k|\mathbf{m})} \right] + \mathbb{E}_{q_{\phi_k}(s_k|\mathbf{m})} [p(\mathbf{m}|s_1, \dots, s_N)] \quad (46)$$

$$= \mathbb{E}_{q_{\phi_k}(s_k|\mathbf{m})} \left[\log \frac{p(s_k)}{q_{\phi_k}(s_k|\mathbf{m})} \right] + \|\mathbf{m} - \frac{1}{N} \sum_j^N \mathbf{s}_j\| \quad (47)$$

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

As both the prior $p(s_k)$ and the approximate posterior $q_{\phi_k}(s_k|\mathbf{m})$ are parametrized by deep neural networks, the gradient of Equation (47) is tractable with respect to the mini-batch.

Choosing a posterior

With the optimization regime set up, we have to choose a fitting posterior distribution and the deep neural network which computes its parameters from data. In many typical VAE settings, the prior density is a fixed probability distribution, typically a multivariate normal distribution and accordingly the approximate posterior also is set to be a normal distribution. As our prior is a greatly complex distribution estimated by the flow, we are not incentivized by this to choose the posterior accordingly. Nevertheless

In the case of the **real musical data** we can not construct any informed posterior distribution, as the signals do not follow any constrained prior form. For the lack of an informed decision and the computational ease of the reparametrization trick, we resort to a Gaussian posterior. More specifically we propose employing a **truncated normal distribution**. The latent sound values are restricted to $[-1, 1]$ and with a truncated Gaussian we can restrict the support to this interval. The reparametrization trick is still applicable, for a concrete description of the sampling refer to ?? . The normal distribution is parametrized by mean and variance $\eta = \{\sigma, \mu\}$.

Choosing an Encoder

The inference model Encoder $\phi(\mathbf{m})$ infers the parameters for the chosen posterior distribution from data. The model takes the time-domain signals as an input, therefore we use the architectural design choices as with the prior networks and employ a non-causal WaveNet architecture. As laid out before this model is fully convolutional, not autoregressive, as the prediction of the parameters is a one-to-one mapping, where each time-point is mapped to a set of parameters. As it is practice we use the same network head for both parameters and append after the last skip-connection output of the WaveNet two convolutional layers with the final one mapping to the two separate parameters. The WaveNet used here is fairly similar to the modules used in the prior flows. They employ the gated convolution and have fixed kernel sizes of 3.

We note that we fix the encoder to work in the time-domain. We propose this to have access to the phase information of the original signals during the mapping $\mathbf{m} \rightarrow s_k$. This does not make it necessary, that the prior models are also working in the time-domain. The samples taken from the approximate posterior during the training of the inference model can be transformed with the same spectral transformation used to train the prior models. We can compute the gradient of the likelihood from the prior under the spectral transformation⁶⁷. Because we are not using the generative priors to generate in this setting, the phase information lost through the transformation

⁶⁷ The torchaudio library provides implementations of real-valued spectrograms and a mel-transform that allow for automatic differentiation.

is not disadvantageous and still is accessible to the inference model.

As we have N different sources with each a trained prior network we will also train N inference networks each with a separate approximate posterior.

Difficulties of this method

We want to make the difficulties with this method clear and list them in the following explicitly:

1. *All sources known and trained with prior*
2. The separation capabilities in both approaches completely rely on the quality of the learned priors. The priors are learned unsupervised and unconditional from random cuts of the instruments stems. The learned manifolds are extremely complex and hard to train. There is little prior work on successfully building generative models for sound, without further conditional information.
3. We have to train a prior model for each source. In the sampling case, we concurrently sample from each of these models, in the case of the inference model we have to train an additional model for all sources. Because the separation happens under the mixing condition, at the time of training, inference or sampling all parts are used.
4. *Unstable sampling / training because of many objectives*

De-noising

The task of de-noising is a simpler objective setting but similar to source separation. In de-noising, we want to remove i.i.d. noise ϵ from the observed signal \hat{s} .

De-noising

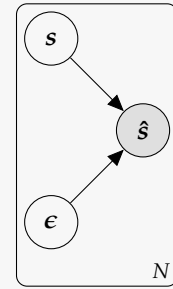


Figure 9: The graphical model for de-noising. The observed signal is generated by the sum of a noise-less signal and a parameter-less sample of Gaussian noise ϵ .

Experiments

As we have laid out in the previous chapter our method mainly depends on the quality of the learned source priors. The generative prior models have to provide an exhaustive and exclusive approximation of the data manifold or its data source. Therefore our experiments strongly focus on analyzing the prior densities under different training regimes and inputs. In general, we have two separate datasets with which we run the experiment, as described in the following section. For each experiment, we use two different data representations. First using the raw time-domain data and second a mel-spectrogram transformation into frequency space. As explained in ?? the frequency representation is lacking phase information which would necessitate a vocoder model to invert the destructive frequency transformation. We keep this step out of scope for this work, as there are also separate successful works on reversing spectrograms to time-domain⁶⁸.

⁶⁸ something about vocoders.

Datasets

We will be working on two datasets. First, a simplified Toy dataset which reduces the problem to a minimal canonical one and second the musdb18 dataset which is a widely-used test-set for musical source separation.

We fix the sampling rate of both datasets to 16.000Hz. For both the Toy Data and the musdb18 dataset we experiment with prior models on time-domain and spectral-domain input.

ToyData

We simplify the problem domain to create a toy-like dataset. We randomly generate waves from four simple oscillations, see Figure 10. Given a wave from each source, the mix is computed by simply taking the mean. When sampling from each source we randomly select a period and phase. The frequencies are restricted to the frequency bounds of the 88 keys of an equal-temperament tuned piano. In our experiments we are gonna model these sources with probability densities, looking especially at the square that will pose a problem, as those only consist of two unique values (-1 and 1). This collapsing posterior would simplify the problem too much, therefore we also vary the amplitude of the sampled signals in the uniform range



Figure 10: One period of each of the four toy sources: sinus, sawtooth, square and triangle wave.

[0.8, 1.0].

In ++later++ section we show that estimating densities over these waves is not giving a smooth manifold. Or differently: in the latent space we can not interpolate between two signals, because the model models, simply put, the sample waves as spiked Dirac deltas.

musdb18

Further we use the *musdb18*⁶⁹ dataset published for the 2018 Signal Separation Evaluation Campaign⁷⁰. The dataset consists of 150 full songs covering various artists and genres, split into train and test sets sized 100 and 50, respectively. Each track is separated into the four sources *drums*, *bass*, *vocals* and *others*. The *others* source channel contains any remaining set of instruments not categorized under the first three ones. The song files are provided in the Stem audio file format⁷¹ and encoded at 44.1kHz. Stem here is terming the provided source channels, we use the terms interchangeably.

The dataset consists of the 100 tracks taken from the DSD100⁷² dataset and 46 track out of the MedleyDB⁷³ dataset. Both dataset contain tracks from a wide range of (western) musical genres, Rock, Pop, Rap, Jazz and Metal. While the upstream data sources provide more fine-grained meta-data, *musdb18* brings all tracks into the same format and stem splitting as described above.

Next to the separated stems, the dataset provides the original (studio) mix for each song. This mix is not equivalent to the simple linear mixing which we get by taking the mean. Nevertheless, the provided mix diverges only insignificantly from an auto-generated mix, as the original sources are provided in their post-compression, post-mixed form. This means that we can use the original mix and assume it to be reasonably close to the canonical linear mix.

As the songs are real natural songs, they are of different lengths. Our models will, in difference, to many other recent methods, not be auto-regressive. Thus we sample fixed-length cuts from the songs as training and test samples. For the *musdb18* data, no pre-processing is applied, as the data already contains the wanted level variability, it spanning different genres and artists.

It is noted that the *musdb18* dataset while providing a remarkable diverse 10hr of real-world music, is a rather small numbered set of samples. Any data modelling from this set of data will consequently be biased. The dataset specifically is created for the accompanying separation challenge and will not translate to general music modelling.

Prior architectures

We construct four distinct prior models for the two datasets and input data representations, respectively. All models follow the flow architecture described in *Method*. Table 1 gives a summary of the networks. Remember for the time-domain priors the affine trans-

⁶⁹ Rafii et al., *MUSDB18 - a Corpus for Music Separation*, 2017.

⁷⁰ Stöter et al., “The 2018 Signal Separation Evaluation Campaign”, 2018.

⁷¹ Native Instruments, *Stem Audio Format*, n.d.

⁷² Liutkus et al., “The 2016 Signal Separation Evaluation Campaign”, 2017.

⁷³ Bittner et al., “MedleyDB 2.0: New Data and a System for Sustainable Data Collection”, 2016.

	blocks	flows	layers	kernel size	features
toy (time)	4	6	10	3	32
musdb18 (time)	8	6	10	3	48
toy (mel)	-	-	-	-	-
musdb18 (mel)	4	32	3	3	512

formations are parametrized by WaveNets, consisting of multiple one-dimensional convolutions. For the spectrogram priors, each is parametrized by un-dilated two-dimensional convolutions as described before. Because the complexity of the coupling layer is smaller in this case, we increase the number of flows per block⁷⁴. In both cases the last layer is initialized to zero, reducing each affine coupling layer to the identity at the beginning of training. The number of features in each convolutional layer is fixed throughout the whole network, as given in the table. All kernels are size 3 or 3×3 , respectively.

Training the priors

Training of the priors is done in mini-batches with an Adam optimizer⁷⁵. The initial learning rate is set to $1e - 4$. The learning rate is decreased with $\gamma = 0.6$ and a fixed five step decrease schedule.

We fine-tune the noise-less flow models by continuing the training with added Gaussian noise on the input samples. Fine-tuning⁷⁶ the noise-conditioned model instead of retraining the model is considerably cheaper and the noised distribution we want to approximate is close to the noise-free distribution of the initial training. We find that fine-tuning converges quickly at $\gamma = 0.1$. We

Testing the priors

Training of the normalizing flow model combines the two objectives: First, we maximize the log-determinant of the *non-volume preserving* affine coupling and the activation normalization layers. This guarantees us the invertibility of these transformations. Second, we maximize the likelihood of the projected samples under the chosen target distribution⁷⁷. For the evaluation, we are interested in the mean average likelihood under the target distribution for different in- and out-of-distribution inputs. Remember that the flow gives a bijective mapping between the input and target space⁷⁸. For an input of size $x \in \mathbb{R}^{1 \times L}$ the flow returns a latent $z \in \mathbb{R}^{1 \times L}$ of the same size. Each time-point latent z_t depends on the inputs from the receptive field of the flow before and after the respective input x_t but is evaluated under the independent margin $z_t \sim \mathcal{N}(0, 1)$. Because therefore the factorized multivariate Gaussian prior is equivalent to a set of one-dimensional Gaussians, the flow is also not size restricted. All transformations in the network are convolutional and use appropriate padding. L can be any chosen length. We want to set the input length larger than the receptive field of the network.

Table 1: The network architectures for the four prior models.

⁷⁴ Kingma and Dhariwal, “Glow: Generative Flow with Invertible 1x1 Convolutions”, 2018.

⁷⁵ Kingma and Ba, “Adam: A Method for Stochastic Optimization”, 2017.

⁷⁶ yosinskiHow2014.

⁷⁷ The target distribution is the prior to our prior model. It is set to a factorized Gaussian.

⁷⁸ necessitated by the invertibility

The latents on either end side of the sample will be based on the larger amount of padded input, while the center values from the sample receive the full receptive field. The size of the receptive field is regimented by the depth and kernel sizes used in the convolutions of the coupling layers and the number of squeeze operations in the blocks (see Figure 8). For an exact derivation of the receptive field size see [Appendix recp size](#).

We show an example of sampling from the toy data priors in time-domain in ?? . It should be unsurprising that the drawn samples are not returning a consistent curve. Each independent latent is connected to a local output defined by the receptive field. As all latents are i.i.d. the generative process, generates locally consistent curves.

De-noising

De-noising is the

In this ablation study we want to check how resilient the generative prior models are to noisy samples. It is hypothesized that the latent space of the model consists of, intuitively, singular peaks at mappings of in-distribution samples⁷⁹. For the separation to be able to optimize under the prior density, we need likelihood to slowly decrease when moving away from positive samples. If the distribution is too peaked only noiseless inputs would be highly likely and adding noise quickly makes samples too unlikely. We propose adding Gaussian noise with a random variance to the positive samples during training. This should smooth out the modeled density.

⁷⁹ Jayaram and Thickstun, “Source Separation with Deep Generative Priors”, 2020.

Testing cross-likelihood

In the full model each prior is used to extract its source channel separately. Through their training they explicitly contract the density for the positive, in-class examples. During separation the priors therefore encounter negative, out-of-distribution samples for the first time. To be useful for separation, it is important that the priors give a low likelihood to samples from the other classes of the dataset.

We test for this by calculating the mean log likelihood of the the test data set under each prior, for each source channel separately. What we anticipate is that the samples from the priors training source are of high likelihood while all other sources are low likelihood. In ?? we show the source dependent likelihoods for the Toy Data. The in-distribution samples are all of high likelihood while all out-of distribution samples are highly un-likely. The Flow model therefore was able to detect out-of-distribution samples while only being trained on in-distribution samples⁸⁰. The estimated densities are discriminative.

⁸⁰ Something psychological research has shown, humans being able to do.

When running the same experiment for the musdb18 the discriminative power does not hold up, see ?? . All signal sources are about equally likely under each prior density. We hypothesize this stems from the fact that the real musical data is severely more com-

plicated compared to the Toy Data. The Flows model the appearance of sound in general, as the in-class variability of sounds is already high, without knowing anything about the *discriminative* differences between the instruments and does not infer the distribution from those features.

Even above that the results exhibit an additional failing for the musdb18 dataset. The third stem in the dataset files *other* is filled with a smorgasbord of different instruments. With our training regime we are implying we can model the sound this set of unrelated instruments with only in-distribution samples with a resulting distribution which is discriminative against another set of unrelated instruments. Intuitively this already fails to be reasonable. Because of this the results show the unintuitive result that the test samples from the *other* class are *less likely* under its own prior model compared to the out-of-distribution samples. While this makes it rather final that the at hand dataset is not completely suitable to our trainings regime, it is a data-dependent restriction that would not correspond to our hypothesized application.

Similar results were shown before in

cite some OOD VAE stuff

The results for both datasets are similar for all tested model architectures and input domains. See Appendix

add results

for the full results.

Increasing discriminative power

With the prior models being not discriminative, it is not possible to train the separation model. Following we will try extend the priors to increase their discriminative power. Above that we propose that our method still holds merit given these results, further work on generative models might result in more discriminative generative models. The method can use any density as a drop-in replacement for the prior in the separation model.

As layed out in the bad out-of-distribution detection of generative models is not a surprising result and multiple recent works go about solving this. We want to increase the

Testing the posteriors

TODO EXPERIMENTS

1. Adding noise better likelihood
2. De-noising single signals

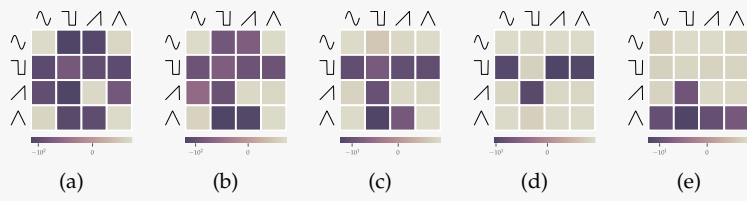


Figure 11: the caption

Bibliography

- Bittner, Rachel M., Julia Wilkins, Hanna Yip, and Juan P. Bello (2016). “MedleyDB 2.0: New Data and a System for Sustainable Data Collection”. In: *ISMIR Late Breaking and Demo Papers* (cit. on p. 30).
- 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] (cit. on p. 12).
- 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 (cit. on p. 16).
- 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] (cit. on p. 18).
- Dauphin, Yann N., Angela Fan, Michael Auli, and David Grangier (2017). “Language Modeling with Gated Convolutional Networks”. In: arXiv: 1612.08083 [cs] (cit. on p. 18).
- 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] (cit. on p. 18).
- Défossez, 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 (cit. on p. 18).
- Dinh, Laurent, David Krueger, and Yoshua Bengio (2015). “NICE: Non-Linear Independent Components Estimation”. In: arXiv: 1410.8516 [cs] (cit. on p. 12).
- Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio (2017). “Density Estimation Using Real NVP”. In: arXiv: 1605.08803 [cs, stat] (cit. on pp. 12, 24).
- 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] (cit. on p. 18).
- 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: (cit. on p. 12).
- Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long Short-Term Memory”. In: *Neural Computation* 9.8, pp. 1735–1780 (cit. on p. 15).
- ISO/TC 43 Acoustics (1975). *ISO 16 Acoustics Standard Tuning Frequency* (cit. on p. 14).
- 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* (cit. on p. 17).

- Jayaram, Vivek and John Thickstun (2020). “Source Separation with Deep Generative Priors”. In: arXiv: [2002.07942 \[cs, stat\]](#) (cit. on pp. [25](#), [32](#)).
- 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 (cit. on p. [10](#)).
- 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 \[cs, eess\]](#) (cit. on p. [15](#)).
- Kaspersen, Esbern Torgard (2019). “HydraNet: A Network For Singing Voice Separation”. In: (cit. on p. [18](#)).
- 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. [16](#), [21](#), [22](#)).
- Kingma, Diederik P. and Jimmy Ba (2017). “Adam: A Method for Stochastic Optimization”. In: arXiv: [1412.6980 \[cs\]](#) (cit. on pp. [24](#), [31](#)).
- Kingma, Diederik P. and Prafulla Dhariwal (2018). “Glow: Generative Flow with Invertible 1x1 Convolutions”. In: arXiv: [1807.03039 \[cs, stat\]](#) (cit. on pp. [12](#), [22](#), [31](#)).
- Kingma, Diederik P. and Max Welling (2014). “Auto-Encoding Variational Bayes”. In: arXiv: [1312.6114 \[cs, stat\]](#) (cit. on p. [11](#)).
- 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 (cit. on p. [14](#)).
- Liutkus, Antoine, Fabian-Robert Stöter, Zafar Rafii, Daichi Kitamura, Bertrand Rivet, Nobutaka Ito, Nobutaka Ono, and Julie Fontecave (2017). “The 2016 Signal Separation Evaluation Campaign”. In: *Latent Variable Analysis and Signal Separation - 12th International Conference, LVA/ICA 2015, Liberec, Czech Republic, August 25-28, 2015, Proceedings*. Ed. by Petr Tichavský, Massoud Babaie-Zadeh, Olivier J.J. Michel, and Nadège Thirion-Moreau. Cham: Springer International Publishing, pp. 323–332 (cit. on p. [30](#)).
- 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 \[cs, eess\]](#) (cit. on p. [17](#)).
- 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 \[cs, eess, stat\]](#) (cit. on p. [18](#)).
- Native Instruments (n.d.). *Stem Audio Format* (cit. on p. [30](#)).
- Neal, Radford M. (2012). “MCMC Using Hamiltonian Dynamics”. In: arXiv: [1206.1901 \[physics, stat\]](#) (cit. on p. [13](#)).
- 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 \[cs\]](#) (cit. on p. [15](#)).
- 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 pp. [16](#), [21](#)).
- Pulse Code Modulation (PCM) of Voice Frequencies* (1972). G.711. ITU-T (cit. on pp. [14](#), [15](#)).
- Rafii, Zafar, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimitakis, and Rachel Bittner (2017). *MUSDB18 - a Corpus for Music Separation*. Version 1.0.0 (cit. on p. [30](#)).
- Rethage, Dario, Jordi Pons, and Xavier Serra (2018). “A Wavenet for Speech Denoising”. In: arXiv: [1706.07162 \[cs\]](#) (cit. on p. [17](#)).

- Rezende, Danilo Jimenez and Shakir Mohamed (2016). “Variational Inference with Normalizing Flows”. In: arXiv: [1505.05770 \[cs, stat\]](#) (cit. on p. 12).
- Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). “Stochastic Backpropagation and Approximate Inference in Deep Generative Models”. In: arXiv: [1401.4082 \[cs, stat\]](#) (cit. on p. 12).
- 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 \[cs, eess\]](#) (cit. on p. 18).
- 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) (cit. on p. 18).
- Stöter, Fabian-Robert, Antoine Liutkus, and Nobutaka Ito (2018). “The 2018 Signal Separation Evaluation Campaign”. In: arXiv: [1804.06267 \[cs, eess\]](#) (cit. on p. 30).
- 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 \[cs\]](#) (cit. on pp. 15, 18).
- Tabak, Esteban and Cristina V. Turner (2013). “A family of nonparametric density estimation algorithms”. In: *COMMUN. PURE & APPL. MATHS.* 66.2, pp. 145–164 (cit. on p. 12).
- 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 \[cs\]](#) (cit. on p. 15).
- 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 \[cs\]](#) (cit. on p. 15).
- 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 (cit. on p. 16).
- Welling, Max and Yee Whye Teh (2011). “Bayesian Learning via Stochastic Gradient Langevin Dynamics”. In: *Proceedings of the 28th International Conference on Machine Learning*. ICML’11. Bellevue, Washington, USA: Omnipress, pp. 681–688 (cit. on pp. 20, 25).