Unsupervised source separation with deep priors

Maurice Frank

July 8, 2020

Contents

Abstract

Research Question

Source separation is the task of finding a set of latent sources $\boldsymbol{s} = [\boldsymbol{s}_1, \dots \boldsymbol{s}_k, \dots, \boldsymbol{s}_N]^T \in \mathbb{R}^{N \times T}$ to an observed mix of those sources $\boldsymbol{m} \in \mathbb{R}^{1 \times T}$. The induced model proposes a mixing function $\boldsymbol{m} = f(\boldsymbol{s})$. The task is to find an approximate inverse model $g(\cdot)$ which retrieves $\boldsymbol{s}$:

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

$$g(\boldsymbol{m}) \approxeq \boldsymbol{s} \tag{2}$$

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 $(\boldsymbol{m}, \boldsymbol{s})$ 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 n 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 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 $\boldsymbol{x} \in \mathcal{D}$ for which there exists an unknown data probability distribution $p^*(\mathcal{D})$. We collect a data set $\{\boldsymbol{x}_1 \dots \boldsymbol{x}_N\}$ with N samples. Further we introduce an approximate model with the density[2] $p_{\boldsymbol{\theta}}(\mathcal{D})$ and model parameters $\boldsymbol{\theta}$. Learning or modeling means finding the values for $\boldsymbol{\theta}$ which will give the closest approximation of the true underlying process:

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

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

The model $p_{\boldsymbol{\theta}}$ 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* biases[3] about what density we

[3] called *inductive biases*

can model, even before we maximize a learning objective using the parameters $\boldsymbol{\theta}$.

In the following described models we assume the sampled data points $\boldsymbol{x}$ to be drawn from $\mathcal{D}$ *independent and identically distributed*[4]. Therefore we can write the data log-likelihood as:

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

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

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 $\boldsymbol{z}$, 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.

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

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

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z} \tag{5}$$

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

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) p(\boldsymbol{z}) d\boldsymbol{z} \tag{6}$$

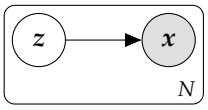

Figure 1: The graphical model with a introduced latent variable $\boldsymbol{z}$. Observed variables are shaded.

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

If the latent is small, discrete, it might be possible to directly marginalize over it. If for example, $\boldsymbol{z}$ is a discrete random variable and the conditional $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ is a Gaussian distribution than the data model density $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ 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 $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ as well as the model posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ are intractable because of the integration over the latent $\boldsymbol{z}$ 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 $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, also called the *inference model*. Again the choice of the model here carries inductive

[6] Jordan et al., "An Introduction to Variational Methods for Graphical Models", 1999.

biases, as such that even in asymptotic expectation we can not obtain the true posterior.

Following the derivation in Kingma and Welling (2019, p. 20) we introduce the inference model into the data likelihood [7]:

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

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

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

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

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

[7] The first step is valid as $q_{\boldsymbol{\theta}}$ is a valid density function and thus integrates to one.

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 approximation [8]. 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):

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

[8] 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.

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

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

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

$$= -\mathbb{D}_{\mathrm{KL}}[q_{\boldsymbol{\phi}}(z|\boldsymbol{x})\|p(z)] + \mathbb{E}_{q_{\boldsymbol{\theta}}(z|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|z)\right] \tag{15}$$

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 latent [9].

The optimization of the $\mathrm{ELBO}_{\boldsymbol{\theta},\boldsymbol{\phi}}$ allows us to jointly optimize the parameter sets $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. The gradient with respect to $\boldsymbol{\theta}$ can be estimated with an unbiased Monte Carlo estimate using data samples [10]. Though we can *not* do the same for the variational parameters $\boldsymbol{\phi}$, as the expectation of the ELBO is over the approximate posterior which depends on $\boldsymbol{\phi}$. By a change of variable of the latent variable we can

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

[10] $\nabla_{\boldsymbol{\theta}} \mathrm{ELBO}_{\boldsymbol{\theta},\boldsymbol{\phi}} \approxeq \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z})$

make this gradient tractable, the so called *reparameterization trick*[11]. We express the $z \sim q_\theta$ as an random sample from a unparametrized source of entropy ϵ and a parametrized transformation:

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

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 $\eta = \{\mu, \sigma\}$.

[11] Kingma and Welling, "Auto-Encoding Variational Bayes", 2014.

The VAE framework

The variational autoencoder (VAE) [11][12] is the previously introduced ideas together into a trainable model.

more

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

Algorithm 1: Training's procedure for a variational autoencoder

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

The β-VAE[13] 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 representations[13]. Using such a constraint is similar to the use of the information bottleneck[14].

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

[14] Burgess et al., "Understanding Disentangling in Beta-VAE", 2018.

Flow based models

Another class of common deep latent models is based on *normalizing* flows[15]. A normalizing flow is a function $f(\boldsymbol{x})$ that maps the input density to a fixed, prescribed density $p(\epsilon) = p(f(\boldsymbol{x}))$, in that normalizing the density [16]. They use a flow for the approximate posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$. Again this is commonly set to be a factorized Gaussian distribution.

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

NICE[17] - volume preserving transformations - coupling layer - triangular shape

Normalizing Flow[18]

RealNVP[19] builds on top of NICE creating a more general, non-volume preserving, normalizing flow.

[15] Tabak and Turner, "A family of non-parametric density estimation algorithms", 2013.

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

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

[18] Rezende and Mohamed, "Variational Inference with Normalizing Flows", 2016.

[19] Dinh, Sohl-Dickstein, et al., "Density Estimation Using Real NVP", 2017.

$$y_{1:d} = x_{1:d} \tag{17}$$

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

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

Glow[20] 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 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.

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

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 signal[21]. Humans sound perception is *stereo* as we have two ears and sense the changes in pressure at two different points in space simultaneously.

[21] 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.

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 Nyquist-Shannon theorem[22] tells us that if the highest frequency in the true signal is B Hz than 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 a evenly spaced grid over the possible value range. More advanced quantization include μ-law encoding[23] in which the quantization intervals are varied with the amplitude[24].

[22] Kotelnikov, "On the Carrying Capacity of the Ether and Wire in Telecommunications", 1933.

[23] *Pulse Code Modulation (PCM) of Voice Frequencies*, 1972.

[24] The actual formula for calculating the μ-law quantization is:

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

As 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[25] has a frequency range of $[27.5\,\text{Hz}, 4186\,\text{Hz}]$ 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.

[25] ISO/TC 43 Acoustics, *ISO 16 Acoustics Standard Tuning Frequency*, 1975.

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 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.

The **WaveNet**[26] introduced an autoregressive generative model for raw audio. It is build upon the similar PixelCNN[27] but adapted for the audio domain. The WaveNet accomplishes this by using dilated

[26] Van den Oord, Dieleman, et al., "WaveNet: A Generative Model for Raw Audio", 2016.

[27] Van den Oord, Kalchbrenner, et al., "Conditional Image Generation with PixelCNN Decoders", 2016,

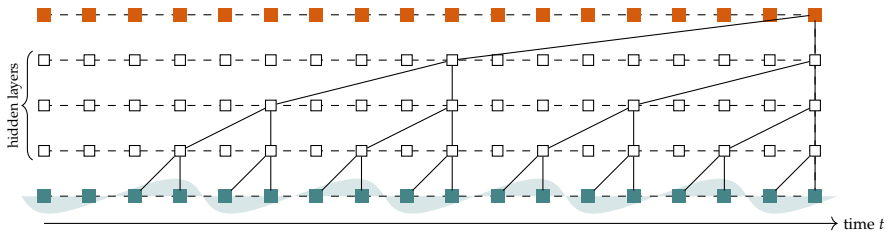

Figure 2: An example of how dilated convolutions are used in the WaveNet. We see three hidden layers with each a kernel size of two. By using the dilations the prediction of the new output element has a receptive field of 18. This convolution is *causal* as the prediction depends only on previous input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

causal convolutions [28]. 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 sturcture of a hidden layer refer to Figure 3. A gated feature, as known from the LSTM[29], 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 easier[30]. The original authors tested the model on multiple audio generation tasks. They used a μ-law encoding[31] which discretizes the range $[-1, 1]$ to allow a set of μtargets 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[32]. This can be alleviated by keeping intermediate hidden activations cached[33]. The WaveNet can be conditioned by adding the weighted conditionals in the gate and feature activations of the gated convolutions[34].

The first generative sounds model to employ a WaveNet is **NSynth**[35]. They construct a VAE where encoder and decoder are both WaveNet-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. (2019) presents another WaveNet-based VAE. They

[28] The *à trous* algorithm (Holschneider et al. (1990)), a common tool in signal processing, uses a Wavelet kernel that is dilated to multiple scales. A dilated convolution, first used in Yu and Koltun (2016) differs in that the convolution operator it-self is *dilated*, having an internal stride.

[29] Hochreiter and Schmidhuber, "Long Short-Term Memory", 1997.

[30] Szegedy et al., "Going Deeper with Convolutions", 2015.

[31] *Pulse Code Modulation (PCM) of Voice Frequencies*, 1972.

[32] Why are they doing that then? The 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.

[33] Paine et al., "Fast Wavenet Generation Algorithm", 2016.

[34] Van den Oord, Kalchbrenner, et al., "Conditional Image Generation with PixelCNN Decoders", 2016.

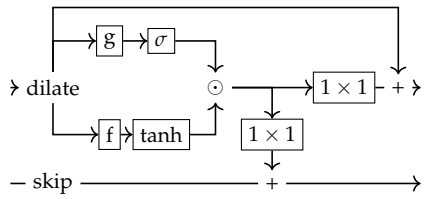

Figure 3: Hidden layer as in the WaveNet. Information flows from left, gets dilated and through the gate and filter. The result gets added to the skip flow and the hidden feature, each with a channel mixer before.

add PixelCNN++?

[35] Kalchbrenner et al., "Efficient Neural Audio Synthesis", 2018.

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 codebook[36] 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[37]

FloWaveNet[38]

[36] Van den Oord, Vinyals, et al., "Neural Discrete Representation Learning", 2017.

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

[38] 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 $p(\boldsymbol{s}|\boldsymbol{m})$ for one source (e.g. extracting only the singing voice out of a mix) or $p(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N|\boldsymbol{m})$ 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. (2018) 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. (2019) adapted a time-domain WaveNet for musical source separation. The non-causal [39] 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 [40].

U-Net based

Jansson et al. (2017) 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

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

[40] ...f a network ... number of hidden layers. Wideness ... number of k... ...res of each of these hidden layers. Making t... deeper sign... ... increases its receptive field.

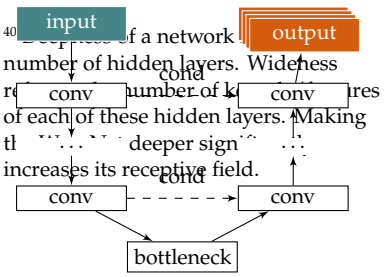

Figure 4: High level idea of a U-Net: the input gets projected into an (informational) bottleneck through some form of convolutional architecture. From this bottleneck the predictions are from an mirrored array of upsampling convolutional layer (either tuples of upsampling and convolutions or through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

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

The Wave-U-Net[41] 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. (2019) 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. (2019) 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.

Kaspersen (2019) 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.

[42]

Demucs[43] also introduces a BiLSTM at the bottleneck[44]. 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 Units[45] 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. (2018) 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 networks[46]).

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

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

[43] Défossez et al., "Demucs: Deep Extractor for Music Sources with Extra Unlabeled Data Remixed", 2019.

[44] Defossez et al., "SING: Symbol-to-Instrument Neural Generator", 2018.

[45] Dauphin et al., "Language Modeling with Gated Convolutional Networks", 2017.

[46] Szegedy et al., "Going Deeper with Convolutions", 2014.

Method

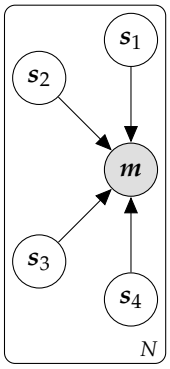

Figure 5: The used graphical model for the source separation task. We have the latent source channel variables s_k. Exemplary here, as in our data, we have four sources. The mix $\boldsymbol{m}$ is observed.

In this chapter we introduce the theoretical idea of our Bayesian modeling of 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 the source separated music 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 6.

The graphical model in Figure 5 implies the following factorization:

$$p(\boldsymbol{m}) = \int^N p(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N, \boldsymbol{m})\, d^N\boldsymbol{s} \tag{20}$$

$$= \int^N p(\boldsymbol{m}|\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N) \cdot p(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N)\, d^N\boldsymbol{s} \tag{21}$$

$$= \int^N p(\boldsymbol{m}|\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N) \cdot p(\boldsymbol{s}_1) \cdots p(\boldsymbol{s}_N)\, d^N\boldsymbol{s} \tag{22}$$

While the conditional $p(\boldsymbol{m}|\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N)$ is not even probabilistic, as the mix is generated from the sources deterministically with the mean, the model posterior $p(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N|\boldsymbol{m})$ is intractable and precisely what we are interested in. For

The general process is as follows:

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 $\mathcal{D}_k$.

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 possi-

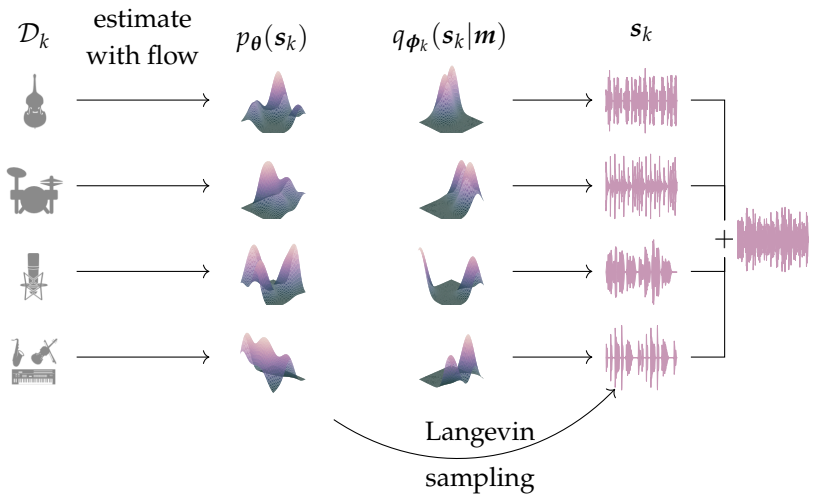

Figure 6: The method visualized

ble. First we can propose a 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). Secondly we can sample directly from the posterior using Stochastic Gradient Langevin Dynamics (SGLD)[47] without explicitly approximating the posterior distribution (sampling setting). Both optimization, either training or sampling, happen under the mixing constraint:

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

$$\sum_{k=1}^{N} s_k = m \tag{23}$$

When using the VAE setting we then can sample from each posterior to retreive the 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 $p_\theta(s_k)$ of the *prior* distribution
2. a parametrized approximate posterior distribution $q_{\phi_k}(s_k|m)$ with parameters η
3. a parametrized *encoder* which gives the parameters for the posterior distribution $\texttt{Encoder}_\phi(m) = \eta$
4. a *decoder* which returns the input m from the latents $\texttt{Decoder}(\{s_1, \ldots, s_N\}) = m$

As stated before the decoder in our case is the mean function, thus not probabilistic and without parameters.

We follow the same steps as previously shown for the latent variable models. First we introduce a 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:

$$\log p(\boldsymbol{m}) = \mathbb{E}^N_{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}\left[\log p(\boldsymbol{m})\right] \tag{24}$$

From here derive the ELBO in the same way as before, just now with N priors instead of one:

$$\begin{aligned}
\mathbb{E}^N_{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}\left[\log p(\boldsymbol{m})\right] &= \mathbb{E}^N_{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}\left[\log \frac{p(\boldsymbol{m}, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_N)}{p(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N|\boldsymbol{m})}\right] && (25)\\
&= \mathbb{E}^N_{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}\left[\log \frac{p(\boldsymbol{m}|\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N) \cdot \prod_k^N p(\boldsymbol{s}_k)}{\prod_k^N q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})} + \log \frac{\prod_k^N q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}{p(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N|\boldsymbol{m})}\right] && (26)\\
&\geq \sum_k^N \mathbb{E}_{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}\left[\log \frac{p(\boldsymbol{s}_k)}{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}\right] + \mathbb{E}_{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}\left[p(\boldsymbol{m}|\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N)\right] && (27)\\
& && (28)
\end{aligned}$$

Like in the VAE we now formulate the lower bound for approximting the graphical model with the approximate posterios $q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})$ which we will parametrize with deep neural networks. The source priors are estimated *a-priori* and independtly and are also parametrized by neural networks. Here we introduce a fairly big assumption, namely that we can model the source distributions independtly from each other when modeling the joint:

$$p(\boldsymbol{m}, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_N) \equiv p(\boldsymbol{m}|\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N) \cdot \prod_k^N p(\boldsymbol{s}_k) \tag{29}$$

It is intuitive that 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. 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.

Finally we note that 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 (24) over those is still correct. The error arising from the independence assumption is captured in the thightness of the ELBO. I would add a paragraph that states what choices need to be made: Encoder, Decoder, Prior

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 representation of the input different network architectures are mode suited. We experiment both with inputing directly the time-domain wave but also modeling 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 invertable transformation parametrized by the other. The RealNVP masks the data spatially or over the channels in two different groups. Glow 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 works exists adapting the architecture specifically to the time-series domain by integrating WaveNet modules, FloWaveNet[48] and WaveGlow[49], respectively.

For the case of the time-domain representation, the data has only one channel (mono-audio). Therefore a Glow-like shuffleing of channels is not possible and we resort to using spatial masking for the coupling layers. In the case of using a spectral representation of the audio data we can also experiment with channel-shuffleing.

blabla

[48] Kim et al., "FloWaveNet : A Generative Flow for Raw Audio", 2019.
[49] Prenger et al., "WaveGlow: A Flow-Based Generative Network for Speech Synthesis", 2018.

wat about waveflow

Fixing the priors

blabla

Modeling the posteriors

Assuming we have now a set of meaningful, discriminative priors $p(\boldsymbol{s}_i)$ we need to combine the independent models to extract sources $\{\boldsymbol{s}_i\}$ given a mix $\boldsymbol{m}$. At the beginning of this section we already introduced the imagined generative model and derived the lower bound for the approximate posterior. Taking a step back again and looking at the posterior we have two general possibilities to retrieve samples from the posterior $p(\boldsymbol{s}_i|\boldsymbol{m})$.

$$p(\boldsymbol{s}_i|m) = \frac{p(\boldsymbol{m}|\boldsymbol{s}_i) \cdot p(\boldsymbol{s}_i)}{p(\boldsymbol{m})} \tag{30}$$

First we can model the approximate posterior $q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})$. By doing so we train a new subsequent function which directly retrieves the predicted sources given a mix. This method is, in principle, the same as we would see in a normal variational-autoencoder. Alternatively, finding the actual posterior to difficult to model, we can sample $\boldsymbol{s}_i \sim p(\cdot|m)$ without computing the actual distribution.

Variational-autoencoder

blabla

Langevin Dynamics

The second option for estimating the posterior $p(\boldsymbol{s}_i|\boldsymbol{m})$ is iteratively sampling from the prior, under the the mixing constraint, using Langevin dynamics[50]. While they are a classical approach to stochastic modeling of a dynamical modelcular system, Welling and Teh (2011) has shown how to employ Langevin dynamics as an MCMC sampling process to estimate the posterior in Bayesian modeling. With Stochastic Gradient Langevin Dynamics (SGLD) we can sample from $\boldsymbol{s}_i \sim p(\cdot|\boldsymbol{m})$ without computing, explicitly, the posterior $p(\boldsymbol{s}_i|\boldsymbol{m})$.

[50] Neal, "MCMC Using Hamiltonian Dynamics", 2012.

explanation

The idea of using SGLD in combination with deep parametrized priors was, concurrently to this work, introduced in Jayaram and Thickstun (2020). The authors argue that direct optimization under the prior distribution is not successfull due to the severe peakedness of the priors. Thus, they argue, the stochasticity, added with the Gaussian noise in SGLD, is needed to not get stuck in local optimas under the manifold. We challenge that explanation as it is, blabla

I like the section so far, but I think you should emphasize more what are the challenges, e.g., for both the VAE approach and the Langevin dynamics we need a prior and a posterior. In addition, since we have a probabilistic approach we need to choose distributions for everything.

Experiments

Datasets

In this section we introduce the two datasets we will be working on. First a simplified Toy dataset which reduces the problem to a minimla canonical one and second the musdb18 dataset which is a widely-used testset for musical source separation.

ToyData

We simplify the problem domain to create a toy-like dataset. We randomly generate waves from four simple oscillations, see Figure 7. 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 a 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 $[0.8, 1.0]$.

Figure 7: One period of each of the four toy sources: sinus, sawtooth, square and triangle wave.

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*[51] dataset published for the 2018 Signal Separation Evaluation Campaign[52]. The datasets consists of 150 full songs covering various artists and genres, splited 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 set of instrument not categorized under the first three ones. The song files are provided in the Stem audio file format[53] and

[51] Rafii et al., *MUSDB18 - a Corpus for Music Separation*, 2017.

[52] Stöter et al., "The 2018 Signal Separation Evaluation Campaign", 2018.

[53] Native Instruments, *Stem Audio Format*, n.d.

encoded at 44.1kHz. Stem here is terming the provided source channels, we use the terms interchangeably.

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 a 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 musdb 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 modeling 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 modeling.

For both the Toy Data and the musdb18 dataset we experiment with prior models on time-domain and spectral-domain input.

Testing the priors

Adding noise

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 samples[54]. 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.

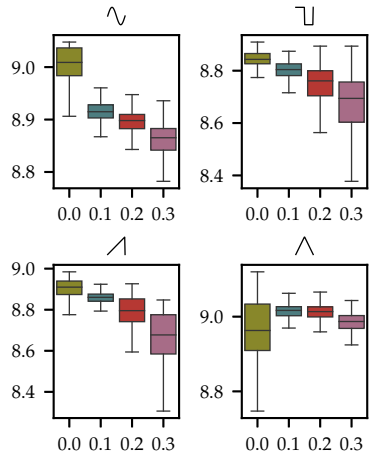

Figure 8: noise likelihood with noise

[54] 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

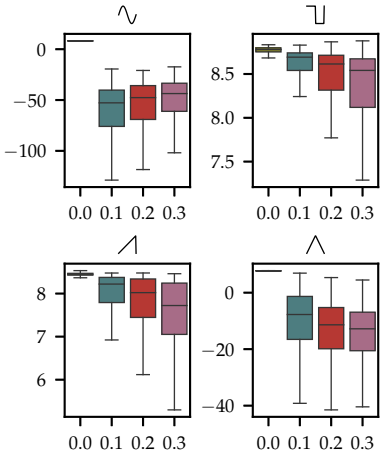

Figure 9: noise likelihood without noise

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 Figure 10 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 samples [55]. The estimated densities are discriminative.

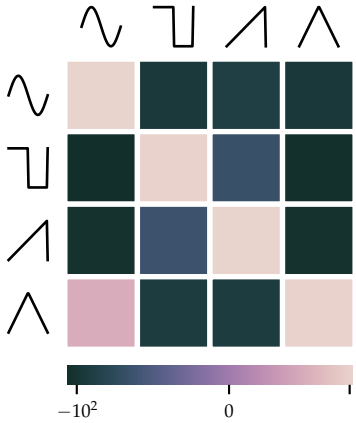

Figure 10: We display the mean average log likelihood of the test data under the different priors and the different signal sources.
[55] Something psychological research has shown, humans being able to do.

When running the same experiment for the musdb18 the discriminative power does not hold up, see Figure 11. 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 complicated 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 out trainings regime, it is a data-dependent restriction that would not correspond to our hypothesized application.

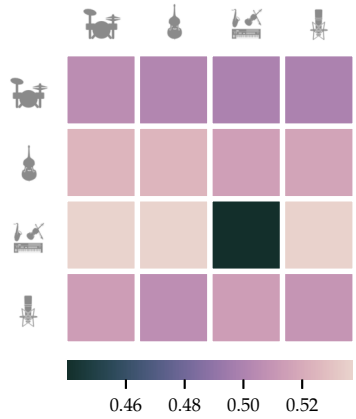

Figure 11: We display the mean average likelihood of the test data under the different priors and the different signal sources.

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 for the full results. add 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

Figure 12: The logits of different classes of the different outputs

Testing the posteriors

Denoising autoencoder

TODO EXPERIMENTS

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

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] (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. 15).

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).

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 (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).

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 p. 12).

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).

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 (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. 23, 26).

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, 22).

Kingma, Diederik P. and Prafulla Dhariwal (2018). "Glow: Generative Flow with Invertible 1x1 Convolutions". In: arXiv: 1807.03039 [cs, stat] (cit. on p. 13).

Kingma, Diederik P. and Max Welling (2014). "Auto-Encoding Variational Bayes". In: arXiv: 1312.6114 [cs, stat] (cit. on p. 12).

Kingma, Diederik P. and Max Welling (2019). "An Introduction to Variational Autoencoders". In: arXiv: 1906.02691 [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. 13).

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. 25).

Neal, Radford M. (2012). "MCMC Using Hamiltonian Dynamics". In: arXiv: 1206.1901 [physics, stat] (cit. on p. 23).

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, 22).

Pulse Code Modulation (PCM) of Voice Frequencies (1972). G.711. ITU-T (cit. on pp. 13, 15).

Rafii, Zafar, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner (2017). *MUSDB18 - a Corpus for Music Separation*. Version 1.0.0 (cit. on p. 25).

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. 25).

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 p. 18).

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 (cit. on p. 15).

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. 14).

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 pp. 14, 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 International Conference on Machine Learning*. ICML'11. Bellevue, Washington, USA: Omnipress, pp. 681–688 (cit. on pp. 20, 23).

Yu, Fisher and Vladlen Koltun (2016). "Multi-Scale Context Aggregation by Dilated Convolutions". In: arXiv: 1511.07122 [cs] (cit. on p. 15).