Unsupervised variational source separation with deep priors

Maurice Frank

June 16, 2020

Contents

Proposal

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

[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.
2. Can we reduce this to an unsupervised setting. Unsupervised relating to the missing pairings of sources and mixes.

Related works

In this chapter, we discuss previous research in supervised and semi-supervised source separation.

Deep Latent-Variable Models

In this section we

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

[3] called *inductive biases*

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

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

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

To form a latent-variable model we introduce a *latent variable*[5]. The data likelihood now is the marginal density of the joint latent density:

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

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

Typically we introduce a factorization of the joint. Most commonly and simplest:

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

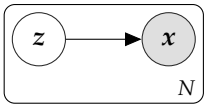

Figure 1: The graphical model with a introduced latent variable $\boldsymbol{z}$. Observed variables are shaded.

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 follow the *variational principle*[6] which introduces an approximate posterior distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, also called the

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

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

test[8]

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

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

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

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

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

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 [9].

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

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

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 [10].

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 [11]. We can *not* though do the same for the variational parameters $\boldsymbol{\phi}$, as the expectation of the ELBO is over the approximate posterior which

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

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

[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.

explain free energy

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

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

depends on $\boldsymbol{\phi}$. By a change of variable of the latent variable we can make this gradient tractable, the so called *reparameterization trick*[12]. We express the $z \sim q_{\boldsymbol{\theta}}$ as an random sample from a unparametrized source of entropy $\boldsymbol{\epsilon}$ and a parametrized transformation:

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

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

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

The VAE framework

VAE [12][13]

The β-VAE[14] 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[14]. Using such a constraint is similar to the use of in the information bottleneck[15].

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

[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.

Flow based models

Another class of common deep latent models is based on *normalizing* flows[16]. 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 [17]. 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[18] - volume preserving transformations - coupling layer - triangular shape

Normalizing Flow[19]

RealNVP[20] builds on top of NICE creating a more general, non-volume preserving, normalizing flow.

[16] Tabak and Turner, "A family of non-parametric 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.

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

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

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

Glow[21] 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

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

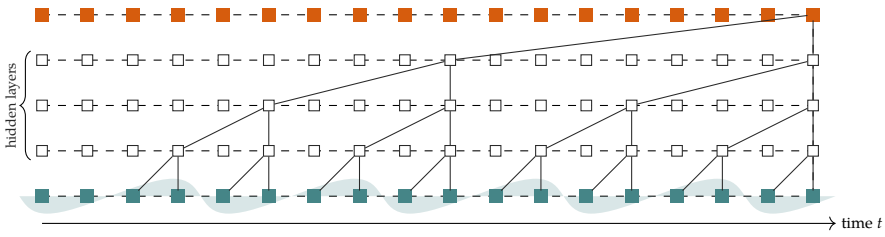

Figure 2: An example of how dilated convolutions are used in the WaveNet. We see three hidden layers with each a kernel size of two. By using the dilations the prediction of the new output element has a receptive field of 18. This convolution is *causal* as the prediction depends only on previous input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

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.

Modelling 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**[22] introduced an autoregressive generative model for raw audio. It is build upon the similar PixelCNN[23] but adapted for the audio domain. The WaveNet accomplishes this by using dilated causal convolutions [24]. 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[25], 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[26]. The original authors tested the model on multiple audio generation tasks. They used a μ-law en-

[22] Van den Oord, Dieleman, et al., "WaveNet: A Generative Model for Raw Audio", 2016.
[23] Van den Oord, Kalchbrenner, et al., "Conditional Image Generation with PixelCNN Decoders", 2016,
[24] The *à trous* alogrithm (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.
[25] Hochreiter and Schmidhuber, "Long Short-Term Memory", 1997.

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

coding[27] 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[28]. This can be alleviated by keeping intermediate hidden activations cached[29]. The WaveNet can be conditioned by adding the weighted conditionals in the gate and feature activations of the gated convolutions (see van den Oord, Kalchbrenner, et al. (2016)).

The first generative sounds model to emply a WaveNet is **NSynth**[30]. 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 predicits 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 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[31] to be most successfull. 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[32]

FloWaveNet[33]

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.

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

[27] *Recommendation G. 711. Pulse Code Modulation (PCM) of Voice Frequencies*, 1988.

[28] 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.

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

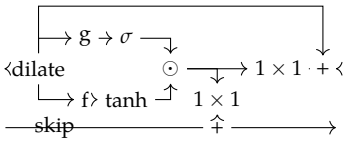

Figure 3: Hidden layer as in the WaveNet. Infomration flows from left, gets dilated and through the gate and filter. The result gets added to the skip flow and the hidden feature, each with a channel mixer before.

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

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

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

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

this case the real valued predicitons bevae better.

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 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[34] 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.

Lluís et al. (2019) adapted a time-domain WaveNet for musical source separation. The non-causal [35] 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 [36].

Demucs[37] is another extension building on the U-Net idea. Having a similar structure to the Wave-U-Net they introduce a bidirectional LSTM at the bottleneck[38]. The LSTM is responsible for keeping long-term temporal informational by running over the high-level latent along the time dimension. They can outperform spectrogram based approaches.

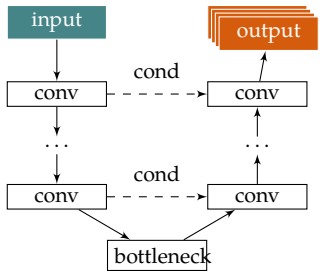

Figure 4: The U-Net

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

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

[36] 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.

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

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

Methodology

Theory

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{m}) = \int \cdots \int \log p(\boldsymbol{m}) \cdot \prod_{k}^{N} q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})\, d^N\boldsymbol{s} \tag{20}$$

$$= \mathbb{E}^{N}_{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})} \left[\log p(\boldsymbol{m})\right] \tag{21}$$

$$= \mathbb{E}^{N}_{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})} \left[\log \frac{p(\boldsymbol{m}, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_N)}{p(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N|\boldsymbol{m})}\right] \tag{22}$$

$$= \mathbb{E}^{N}_{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})} \left[\log \frac{p(\boldsymbol{m}|\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N) \cdot \prod_k^N p(\boldsymbol{s}_k)}{\prod_k^N q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})} + \log \frac{\prod_k^N q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}{p(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N|\boldsymbol{m})}\right] \tag{23}$$

$$\geq \sum_{k}^{N} \mathbb{E}_{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})} \left[\log \frac{p(\boldsymbol{s}_k)}{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})}\right] + \mathbb{E}_{q_{\boldsymbol{\phi}_k}(\boldsymbol{s}_k|\boldsymbol{m})} \left[p(\boldsymbol{m}|\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N)\right] \tag{24}$$

$$\tag{25}$$

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

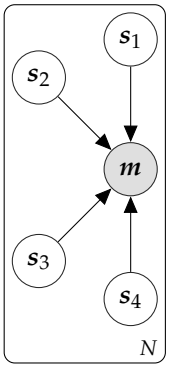

Figure 5: One period of each of the four toy sources: sinus, sawtooth, square and triangle wave.

Datasets

ToyData

We simplify the problem domain to create a toy-like dataset. We randomly generate waves from four simple oscillations, see Figure 5. 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-temprament tuned piano. In our experiments we are gonna model these sources with probability density, 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 ++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*[39] dataset published for the 2018 Signal Separation Evaluation Campaign[40]. The datasets contains 150 full songs covering various artists and genres, splitted into train and test set 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 three first ones. The song files are provided in the Stem audio file format[41] and encoded at 44.1kHz. The stem in the file are the source channels, we use the terms interchangeably.

Next to the separate 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 kanonical 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

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

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

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

spanning different genres and artists.

It is noted that the musdb18 dataset, while proving 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 accompying separation challenge and will not translate to general music modeling.

Planning

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. 8).

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. 10).

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. 11).

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. 11).

Dinh, Laurent, David Krueger, and Yoshua Bengio (2015). "NICE: Non-Linear Independent Components Estimation". In: arXiv: 1410.8516 [cs] (cit. on p. 8).

Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio (2017). "Density Estimation Using Real NVP". In: arXiv: 1605.08803 [cs, stat] (cit. on p. 8).

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. 8).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Computation* 9.8, pp. 1735–1780 (cit. on p. 9).

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. 9).

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. 11).

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 pp. 6, 7).

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. 10).

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 p. 10).

Kingma, Diederik P. and Prafulla Dhariwal (2018). "Glow: Generative Flow with Invertible 1x1 Convolutions". In: arXiv: 1807.03039 [cs, stat] (cit. on p. 8).

Kingma, Diederik P. and Max Welling (2014). "Auto-Encoding Variational Bayes". In: arXiv: 1312.6114 [cs, stat] (cit. on p. 8).

Kingma, Diederik P. and Max Welling (2019). "An Introduction to Variational Autoencoders". In: arXiv: 1906.02691 [cs, stat] (cit. on p. 7).

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. 11).

Native Instruments (n.d.). *Stem Audio Format* (cit. on p. 12).

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. 10).

Prenger, Ryan, Rafael Valle, and Bryan Catanzaro (2018). "WaveGlow: A Flow-Based Generative Network for Speech Synthesis". In: arXiv: 1811.00002 [cs, eess, stat] (cit. on p. 10).

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. 12).

Recommendation G. 711. Pulse Code Modulation (PCM) of Voice Frequencies (1988) (cit. on p. 10).

Rethage, Dario, Jordi Pons, and Xavier Serra (2018). "A Wavenet for Speech Denoising". In: arXiv: 1706.07162 [cs] (cit. on p. 10).

Rezende, Danilo Jimenez and Shakir Mohamed (2016). "Variational Inference with Normalizing Flows". In: arXiv: 1505.05770 [cs, stat] (cit. on p. 8).

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. 8).

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. 11).

Stöter, Fabian-Robert, Antoine Liutkus, and Nobutaka Ito (2018). "The 2018 Signal Separation Evaluation Campaign". In: arXiv: 1804.06267 [cs, eess] (cit. on p. 12).

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. 9).

Tabak, Esteban and Cristina V. Turner (2013). "A family of nonparametric density estimation algorithms". In: *COMMUN. PURE & APPL. MATHS.* 66.2, pp. 145–164 (cit. on p. 8).

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. 9).

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. 9, 10).

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. 10).

Yu, Fisher and Vladlen Koltun (2016). "Multi-Scale Context Aggregation by Dilated Convolutions". In: arXiv: 1511.07122 [cs] (cit. on p. 9).