

*Unsupervised variational source separation with deep priors*

*Maurice Frank*

*June 22, 2020*



# Contents

<b>Abstract</b>	<b>5</b>
<b>Research Question</b>	<b>7</b>
<b>Background</b>	<b>9</b>
<b>Related work</b>	<b>15</b>
<b>Theory</b>	<b>17</b>
Modeling the priors . . . . .	18
Modeling the posteriors . . . . .	19
<b>Experiments</b>	<b>21</b>
Datasets . . . . .	21
Testing the priors . . . . .	22



## *Abstract*



## Research Question

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

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

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

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

<sup>1</sup> For the setting of music think of the classes being  $\{\text{guitar}, \text{piano}, \text{voice}, \dots\}$

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.





# Background

In this section we In general this research work is build on prior work first on deep generative models, or latetn models, which try to estimate the distribution / density of a data space to find smooth manifolds. Second we will give an overview of contemporary works on using deep models for music and speech analysis and generation.

## Deep Latent-Variable Models

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

For our process, we have observations from the data space  $x \in \mathcal{D}$  for which there exists an unknown data probability distribution  $p^*(\mathcal{D})$ . We collect a data set  $\{x_1 \dots x_N\}$  with  $N$  samples. We introduce an approximate model with density<sup>2</sup>  $p_\theta(\mathcal{D})$  and model parameters  $\theta$ . Learning or modeling means finding the values for  $\theta$  which will give the closest approximation of the true underlying process:

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

The model  $p_\theta$  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<sup>3</sup> about what density we can model, even before we maximize a learning objective using the parameters  $\theta$ .

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

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

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

To form a latent-variable model we introduce a *latent variable*<sup>5</sup>. The

<sup>2</sup> We write density and distribution interchangeably to denote a probability function.

<sup>3</sup> called *inductive biases*

<sup>4</sup> meaning the sample of one datum does not depend on the other data points

<sup>5</sup> Latent variables are part of the directed graphical model but not observed.

data likelihood now is the marginal density of the joint latent density:

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

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

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

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

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

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

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

Following the derivation in Kingma and Welling (2019, p. 20) we introduce the inference model into the data likelihood <sup>7</sup>:

test<sup>8</sup>

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

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

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

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

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

Note that we separated the likelihood into two parts. The second part is the (positive) Kullback-Leibler divergence of the approximate posterior from the true intractable posterior. This unknown divergence states the ‘correctness’ of our approximation <sup>9</sup>.

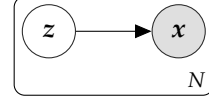


Figure 1: The graphical model with a introduced latent variable  $z$ . Observed variables are shaded.

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

<sup>7</sup> The first step is valid as  $q_{\theta}$  is a valid density function and thus integrates to one.

<sup>8</sup> Jordan et al., “An Introduction to Variational Methods for Graphical Models”, 1999.

<sup>9</sup> 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.

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

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

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

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

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

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

Under this factorization, we separated the lower bound into two parts. First, the divergence of the approximate posterior from the latent prior distribution and second the data posterior likelihood from the latent <sup>10</sup>.

The optimization of the  $\text{ELBO}_{\theta, \phi}$  allows us to jointly optimize the parameter sets  $\theta$  and  $\phi$ . The gradient with respect to  $\theta$  can be estimated with an unbiased Monte Carlo estimate using data samples <sup>11</sup>. We can *not* though do the same for the variational parameters  $\phi$ , as the expectation of the ELBO is over the approximate posterior which depends on  $\phi$ . By a change of variable of the latent variable we can make this gradient tractable, the so called *reparameterization trick*<sup>12</sup>. We express the  $z \sim q_{\theta}$  as an random sample from a unparametrized source of entropy  $\epsilon$  and a parametrized transformation:

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

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

### The VAE framework

VAE <sup>12,13</sup>

The  $\beta$ -VAE<sup>14</sup> extends the VAE objective with an  $\beta$  hyperparameter in front of the KL divergence. The value  $\beta$  gives a constraint on the latent space controlling the capacity of it. Adapting  $\beta$  gives a trade-off between the reconstruction quality of the autoencoder and the simplicity of the latent representations<sup>14</sup>. Using such a constraint is similar to the use of in the information bottleneck<sup>15</sup>.

### Flow based models

Another class of common deep latent models is based on *normalizing flows*<sup>16</sup>. A normalizing flow is a function  $f(x)$  that maps the input

explain free energy

<sup>10</sup> this will later be the reconstruction error. How well can we return to the data density from latent space

<sup>11</sup>  $\nabla_{\theta} \text{ELBO}_{\theta, \phi} \approx \nabla_{\theta} \log p_{\theta}(x, z)$

<sup>12</sup> Kingma and Welling, "Auto-Encoding Variational Bayes", 2014.

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

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

<sup>15</sup> Burgess et al., "Understanding Disentangling in Beta-VAE", 2018.

<sup>16</sup> Tabak and Turner, "A family of non-parametric density estimation algorithms", 2013.

density to a fixed, prescribed density  $p(\epsilon) = p(f(x))$ , in that normalizing the density<sup>17</sup>. They use a flow for the approximate posterior  $q_\phi(z|x)$ . Again this is commonly set to be a factorized Gaussian distribution.

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

NICE<sup>18</sup> - volume preserving transformations - coupling layer - triangular shape

Normalizing Flow<sup>19</sup>

RealNVP<sup>20</sup> builds on top of NICE creating a more general, non-volume preserving, normalizing flow.

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

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

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

Glow<sup>21</sup> extended the RealNVP by introducing invertible  $1 \times 1$ -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.

### 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**<sup>22</sup> introduced an autoregressive generative model for raw audio. It is build upon the similar PixelCNN<sup>23</sup> but adapted for the audio domain. The WaveNet accomplishes this by using dilated causal convolutions<sup>24</sup>. Using a stack of dilated convolutions increases

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

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

<sup>19</sup> Rezende and Mohamed, "Variational Inference with Normalizing Flows", 2016.

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

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

<sup>22</sup> Van den Oord, Dieleman, et al., "WaveNet: A Generative Model for Raw Audio", 2016.

<sup>23</sup> Van den Oord, Kalchbrenner, et al., "Conditional Image Generation with PixelCNN Decoders", 2016,

<sup>24</sup> 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.

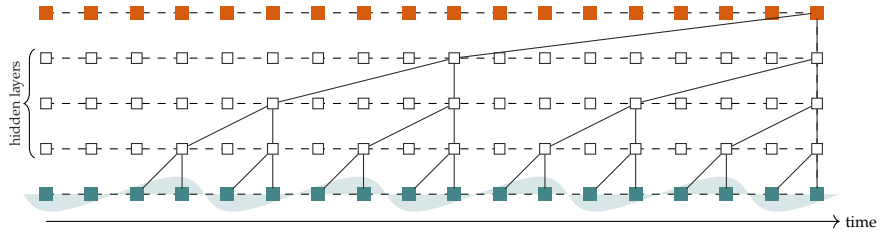


Figure 2: An example of how dilated convolutions are used in the WaveNet. We see three hidden layers with each a kernel size of two. By using the dilations the prediction of the new output element has a receptive field of 18. This convolution is *causal* as the prediction depends only on previous input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

the receptive field of the deep features without increasing the computational complexity, see Figure 2. Further, the convolutions are gated and the output is constructed from skip connections. For the structure of a hidden layer refer to Figure 3. A gated feature, as known from the LSTM<sup>25</sup>, 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<sup>26</sup>. The original authors tested the model on multiple audio generation tasks. They used a  $\mu$ -law encoding<sup>27</sup> which discretizes the range  $[-1, 1]$  to allow a set of  $\mu$  targets and an multi-class cross-entropy training objective. While being quite unnatural this is done to avoid making any assumptions about the target distribution. Sound generation with a WaveNet is slow as the autoregressiveness requires the generation value by value<sup>28</sup>. This can be alleviated by keeping intermediate hidden activations cached<sup>29</sup>. 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 employ a WaveNet is **NSynth**<sup>30</sup>. 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

<sup>25</sup> Hochreiter and Schmidhuber, “Long Short-Term Memory”, 1997.

<sup>26</sup> Szegedy et al., “Going Deeper with Convolutions”, 2015.

<sup>27</sup> Recommendation G. 711. Pulse Code Modulation (PCM) of Voice Frequencies, 1988.

<sup>28</sup> 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.

<sup>29</sup> Paine et al., “Fast Wavenet Generation Algorithm”, 2016.

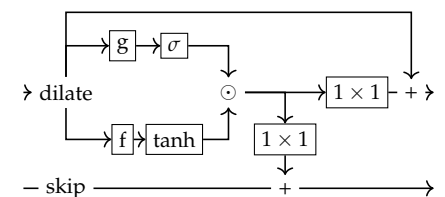


Figure 3: Hidden layer as in the WaveNet. Information flows from left, gets dilated and through the gate and filter. The result gets added to the skip flow and the hidden feature, each with a channel mixer before.

add PixelCNN++?

<sup>30</sup> 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<sup>31</sup> 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<sup>32</sup>

FloWaveNet<sup>33</sup>

<sup>31</sup> Van den Oord, Vinyals, et al., “Neural Discrete Representation Learning”, 2017.

<sup>32</sup> Prenger et al., “WaveGlow: A Flow-Based Generative Network for Speech Synthesis”, 2018.

<sup>33</sup> 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.

ICA???

### Source separation

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

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  $\mu$ -law objective. They show that for their case the real valued predictions behave 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<sup>34</sup> 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

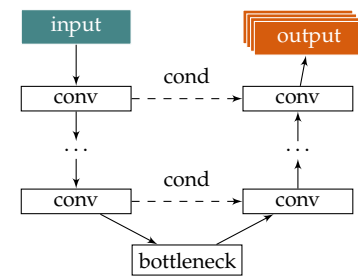


Figure 4: High level idea of a U-Net: the input gets projected into an (informational) bottleneck through some form of convolutional architecture. From this bottleneck the predictions are from an mirrored array of upsampling convolutional layer (either tuples of upsampling and convolutions or through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

<sup>34</sup> Stoller et al., “Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation”, 2018.

source separation. The non-causal <sup>35</sup> 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 <sup>36</sup>.

Demucs<sup>37</sup> 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<sup>38</sup>. The LSTM is responsible for keeping long-term temporal information by running over the high-level latent along the time dimension. They can outperform spectrogram based approaches.

<sup>35</sup> In this setting the WaveNet can be non-causal because the prediction happens from the given mix and is not autoregressive.

<sup>36</sup> 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.

<sup>37</sup> Défossez et al., “Demucs: Deep Extractor for Music Sources with Extra Unlabeled Data Remixed”, 2019.

<sup>38</sup> Defossez et al., “SING: Symbol-to-Instrument Neural Generator”, 2018.



# Theory

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 model the density  $p(s_k|m)$  without using sample tuples  $(m, \{s_1, \dots, s_N\})$ . We follow the same steps as previously shown for the latent variable models. First we introduce an approximate posterior  $q_{\phi_k}(s_k|m)$  for each source channel. Next we express the mix density as the expectation over those posteriors:

$$\log p(m) = \mathbb{E}_{q_{\phi_k}(s_k|m)}^N [\log p(m)] \quad (20)$$

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

$$\mathbb{E}_{q_{\phi_k}(s_k|m)}^N [\log p(m)] = \mathbb{E}_{q_{\phi_k}(s_k|m)}^N \left[ \log \frac{p(m, s_1, \dots, s_N)}{p(s_1, \dots, s_N|m)} \right] \quad (21)$$

$$= \mathbb{E}_{q_{\phi_k}(s_k|m)}^N \left[ \log \frac{p(m|s_1, \dots, s_N) \cdot \prod_k^N p(s_k)}{\prod_k^N q_{\phi_k}(s_k|m)} + \log \frac{\prod_k^N q_{\phi_k}(s_k|m)}{p(s_1, \dots, s_N|m)} \right] \quad (22)$$

$$\geq \sum_k^N \mathbb{E}_{q_{\phi_k}(s_k|m)} \left[ \log \frac{p(s_k)}{q_{\phi_k}(s_k|m)} \right] + \mathbb{E}_{q_{\phi_k}(s_k|m)} [p(m|s_1, \dots, s_N)] \quad (23)$$

$$(24)$$

Like in the VAE we now formulate the lower bound for approximating the graphical model with the approximate posteriors  $q_{\phi_k}(s_k|m)$  which we will parametrize with deep neural networks. The source priors are estimated *a-priori* and independently and are also parametrized by neural networks. Here we introduce a fairly big assumption,

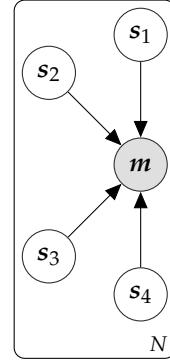


Figure 5: The used graphical model for the source separation task. We have the latent source channel variables  $s_k$ . Exemplary here, as in our data, we have four sources. The mix  $m$  is observed.

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

$$p(\mathbf{m}, s_1, \dots, s_N) \equiv p(\mathbf{m} | s_1, \dots, s_N) \cdot \prod_k^N p(s_k) \quad (25)$$

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  $(s_1, \dots, s_N)$  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 (20) over those is still correct. The error arising from the independence assumption is captured in the tightness of the ELBO.

### *Modeling the priors*

The first step in the training procedure is the training of the independent source priors  $p(s_k)$ . We have to choose a model for estimating the density that results in smooth, tight densities but also is capable enough to capture the complex data space of audio data. We choose to estimate the priors with flow models for which we gave an overview in ???. For different representation of the input different network architectures are more suited. We experiment both with inputting 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 invertible transformation parametrized by the other. The RealNVP masks the data spatially or over the channels in two different groups. Glow learns a  $1 \times 1$ -convolution which permutes the data channel-wise and then just splits the data along the channel dimension into two. For both RealNVP and Glow prior works exist adapting the architecture specifically to the time-series domain by integrating WaveNet modules, FloWaveNet<sup>39</sup> and WaveGlow<sup>40</sup>, respectively.

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

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

<sup>40</sup> Prenger et al., “WaveGlow: A Flow-Based Generative Network for Speech Synthesis”, 2018.

nels 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.

blablabla

*Fixing the priors*

*Modeling the posteriors*

*Variational-autoencoder*

*Langevin Dynamics*

The second option for estimating the posterior  $p(s_i|m)$  is iteratively sampling from the prior, under the the mixing constraint, using Langevin dynamics<sup>41</sup>. 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  $s_i \sim p(\cdot|m)$  without computing, explicitly, the posterior  $p(s_i|m)$ .

<sup>41</sup> 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



# 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 minimal 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 6. 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 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*<sup>42</sup> dataset published for the 2018 Signal Separation Evaluation Campaign<sup>43</sup>. The dataset consists of 150 full songs covering various artists and genres, splitted 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<sup>44</sup> and



Figure 6: One period of each of the four toy sources: sinus, sawtooth, square and triangle wave.

<sup>42</sup> Rafii et al., *MUSDB18 - a Corpus for Music Separation*, 2017.

<sup>43</sup> Stöter et al., "The 2018 Signal Separation Evaluation Campaign", 2018.

<sup>44</sup> 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 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 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 accompying separation challenge and will not translate to general music modeling.

### Testing the priors

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

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

We test for this by calculating the mean log likelihood of the the test data set under each prior, for each source channel separatly. 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 7 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 <sup>45</sup>. The estaimted densities are discriminative.

When running the same experiment for the musdb18 the discriminative power does not hold up, see Figure 8. All signal sources are about equally likely under each prior density. We hypothesize this stems from the fact that the real musical data is severly more compli-

	$\sim$	$\cup$	$\nearrow$	$\wedge$
$\sim$	8.8	-51	-25	-61
$\cup$	-2.2e+02	7.6	-1.4	-1.8e+02
$\nearrow$	-1.4e+02	-0.82	8.8	-1.3e+02
$\wedge$	0.15	-34	-37	7.4

Figure 7: We display the mean average log likelihood of the test data under the different priors and the different signal sources.

<sup>45</sup> Something psychological research has shown, humans being able to do.

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

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

Similar results were shown before in .

The results for both datasets are similar for all tested model architectures and input domains. See Appendix [for the full results](#).

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 discriminative priors. 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 suprising result and multiple recent works go about solving this. We want to increase the

	0.51	0.5	0.5	0.5
	0.52	0.52	0.52	0.52
	0.54	0.54	0.44	0.54
	0.51	0.51	0.51	0.51

Figure 8: We display the mean average likelihood of the test data under the different priors and the different signal sources.

cite some OOD VAE stuff

add results





# 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. 11).
- Chorowski, Jan, Ron J. Weiss, Samy Bengio, and Aaron 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. 13).
- 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. 16).
- 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. 16).
- 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).
- 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. 11).
- Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long Short-Term Memory”. In: *Neural Computation* 9.8, pp. 1735–1780 (cit. on p. 13).
- 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. 12).
- 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. 15).
- Jayaram, Vivek and John Thickstun (2020). “Source Separation with Deep Generative Priors”. In: arXiv: [2002.07942 \[cs, stat\]](#) (cit. on p. 19).

- 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. 13).
- 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. 14, 18).
- Kingma, Diederik P. and Prafulla Dhariwal (2018). “Glow: Generative Flow with Invertible 1x1 Convolutions”. In: arXiv: [1807.03039 \[cs, stat\]](#) (cit. on p. 12).
- Kingma, Diederik P. and Max Welling (2014). “Auto-Encoding Variational Bayes”. In: arXiv: [1312.6114 \[cs, stat\]](#) (cit. on p. 11).
- Kingma, Diederik P. and Max Welling (2019). “An Introduction to Variational Autoencoders”. In: arXiv: [1906.02691 \[cs, stat\]](#) (cit. on p. 10).
- 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. 15).
- Native Instruments (n.d.). *Stem Audio Format* (cit. on p. 21).
- Neal, Radford M. (2012). “MCMC Using Hamiltonian Dynamics”. In: arXiv: [1206.1901 \[physics, stat\]](#) (cit. on p. 19).
- 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. 13).
- 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. 14, 18).
- 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. 21).
- Recommendation G. 711. *Pulse Code Modulation (PCM) of Voice Frequencies* (1988) (cit. on p. 13).
- Rethage, Dario, Jordi Pons, and Xavier Serra (2018). “A Wavenet for Speech Denoising”. In: arXiv: [1706.07162 \[cs\]](#) (cit. on p. 15).
- 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. 11).
- 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. 15).
- Stöter, Fabian-Robert, Antoine Liutkus, and Nobutaka Ito (2018). “The 2018 Signal Separation Evaluation Campaign”. In: arXiv: [1804.06267 \[cs, eess\]](#) (cit. on p. 21).
- 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. 13).
- 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. 11).
- 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. 12).
- 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. 12, 13).
- 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. 14).
- 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 p. 19).
- Yu, Fisher and Vladlen Koltun (2016). “Multi-Scale Context Aggregation by Dilated Convolutions”. In: arXiv: [1511.07122 \[cs\]](#) (cit. on p. 12).