Queen Mary University of London
School of Electronic Engineering and Computer Science

Deep Learning for Music Information Retrieval in Limited Data Scenarios

Daniel Stoller

PhD Thesis

Submitted in partial fulfillment of the requirements
of the Degree of Doctor of Philosophy

July 2020

Statement of Originality

I, Daniel Stoller, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged and my contribution indicated. Previously published material is also acknowledged herein.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

The copyright of this thesis rests with the author.

Signature:

Date: 21/07/2020

Abstract

While deep learning (DL) models have achieved impressive results in settings where large amounts of annotated training data are available, overfitting often degrades performance when data is more limited. To improve the generalisation of DL models, we investigate "data-driven priors" that exploit additional unlabelled data or labelled data from related tasks. Unlike techniques such as data augmentation, these priors are applicable across a range of machine listening tasks, since their design does not rely on problem-specific knowledge.

We first consider scenarios in which parts of samples can be missing, aiming to make more datasets available for model training. In an initial study focusing on audio source separation (ASS), we exploit additionally available unlabelled music and solo source recordings by using generative adversarial networks (GANs), resulting in higher separation quality. We then present a fully adversarial framework for learning generative models with missing data. Our discriminator consists of separately trainable components that can be combined to train the generator with the same objective as in the original GAN framework. We apply our framework to image generation, image segmentation and ASS, demonstrating superior performance compared to the original GAN.

To improve performance on any given MIR task, we also aim to leverage datasets which are annotated for similar tasks. We use multi-task learning (MTL) to perform singing voice detection and singing voice separation with one model, improving performance on both tasks. Furthermore, we employ meta-learning on a diverse collection of ten MIR tasks to find a weight initialisation for a "universal MIR model" so that training the model on any MIR task with this initialisation quickly leads to good performance.

Since our data-driven priors encode knowledge shared across tasks and datasets, they are suited for high-dimensional, end-to-end models, instead of small models relying on task-specific feature engineering, such as fixed spectrogram representations of audio commonly used in machine listening. To this end, we propose "Wave-U-Net", an adaptation of the U-Net, which can perform ASS directly on the raw waveform while performing favourably to its spectrogram-based counterpart. Finally, we derive "Seq-U-Net" as a causal variant of Wave-U-Net, which performs comparably to Wavenet and Temporal Convolutional Network (TCN) on a variety of sequence modelling tasks, while being more computationally efficient.

Acknowledgements

Doing a PhD is a long journey, so you're much better off when taking it together with other people. I was very fortunate to have so many amazing fellow travellers at **C4DM**! Thanks everyone there for creating such an engaging workplace with lots of interesting discussions. In particular, a big thanks to my supervisors Simon, Sebastian and Emmanouil, for all being absolutely fantastic. You were always around to help not just with learning how to do research, but also how to grow as a person as well. And on top of that, discussions with you are always very enjoyable since you are very friendly and down-to-earth. A special thanks goes out to Sebastian for sticking with me even after leaving Queen Mary University – I feel very grateful to be supervised by such a dedicated person committed to raising the next generation of researchers.

Thanks to the organisers of the Media and Arts Technology PhD programme that funded my PhD research[1], in particular Jonathan who helped me with a lot of organisational matters throughout the years.

A big shoutout goes out to all my friends that provided me with support and some time off from the stressful PhD schedule, including Peter, Matt, the D&D role-playing group, and my community of video gamers.

Thanks to my parents that helped me with moving to London and supported me throughout the PhD, also in the form of care packages full of nice German food :) Thanks for imbuing me with kindness, humility and perseverance and enabling me to grow into the person I am today.

Last but certainly not least, thanks Mi for always being there for me, not only emotionally, but also by providing tons of advice and feedback on my research and career goals. Dr. Tian, you are extraordinary.

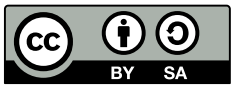
[1]This work was supported by the EPSRC grant EP/L01632X/1.

Contents

List of Figures

List of Tables

Licence



List of abbreviations

ASR	Automatic Speech Recognition
ASS	Audio Source Separation
CNN	Convolutional Neural Network
DFT	Discrete Fourier Transform
DL	Deep Learning
DNN	Deep Neural Network
FFT	Fast Fourier Transform
GAN	Generative Adversarial Network
GMM	Gaussian Mixture Model
GRU	Gated Recurrent Unit
HMM	Hidden Markov Model
LSTM	Long-Short Term Memory network
MIR	Music Information Retrieval
MSE	Mean Squared Error
MTL	Multi-Task Learning
RNN	Recurrent Neural Network
SDR	Signal-to-Distortion Ratio
STL	Single-Task Learning
SVD	Singing Voice Detection
SVS	Singing Voice Separation

Chapter 1

Introduction

1.1 Motivation

The deep learning revolution has initiated an immense amount of progress in many fields where machine learning is applied, such as computer vision [Krizhevsky et al., 2012], robotics [Sünderhauf et al., 2018] and audio processing [Graves et al., 2013]. However, due to the data-driven nature of current DL models, progress has been much more pronounced in settings where large amount of high-quality, labelled data is available, such as object recognition [Krizhevsky et al., 2012, Deng et al., 2009].

In areas where labelled data is more limited due to labelling cost or copyright-related legal restrictions on data sharing, such as medical imaging, music information retrieval or wildlife detection from audio signals [Morfi, 2019], deep learning models tend to overfit the training set and generalise poorly to unseen test data. To improve generalisation, prior knowledge is often incorporated in one or more ways. One way is to use more specialised architectures for deep neural networks (DNNs), such as convolutional neural networks (CNNs), which have a comparatively low number of parameters due to their sparse connectivity pattern between neurons. Another approach involves hand-crafting feature sets that can be used as input to the model instead of the raw sample data. However, with both of the above approaches a glass ceiling is quickly reached: integrating prior knowledge into DNNs is difficult since it is often a challenge to understand what each component of a DNN is doing [Ribeiro et al., 2016], and constructing suitable features requires deep domain knowledge and yields diminishing returns.

Even though data specifically prepared for a certain task is often limited, additional data is usually available that is related to the task in some way. This includes unlabelled data similarly distributed to the task dataset, but also labelled

datasets normally used for other tasks. This also applies to music information retrieval, where unlabelled music is available on very large scales [Bogdanov et al., 2019], and many related tasks exist [IMIRSEL, 2020] such as instrument transcription, melody estimation and beat detection – correlations between the labels from different tasks mean that correctly predicting labels for one task is also helpful for solving the other tasks. At the same time, the potential of multi-task [Caruana, 1998] and transfer learning [Pan and Yang, 2009] to exploit this additional data is still largely untapped in the MIR field.

The immense success of "end-to-end" DL models that process the input directly over approaches based on feature engineering also indicate the superiority of learning features from data over hand-crafting features in an attempt to encode problem knowledge. Compared to fields such as computer vision, where images are processed directly by DNNs, most DL approaches in machine listening still make use of spectral audio features [Sigtia et al., 2015, Grill and Schlüter, 2015, Huang et al., 2014], such as the magnitudes of a Short-Time Fourier Transform. The same applies to models that output audio signals, e.g. audio source separation (ASS) [Uhlich et al., 2015] or audio generation [Vasquez and Lewis, 2019]. In that case, final audio signals have to be recovered from the predicted spectrograms' magnitudes, a process that is only approximate and can produce output artifacts. In light of the above, DNNs that process audio signals in an end-to-end fashion would be desirable. Instead of employing a specific "hand-crafted" prior on how low-level representations of sound should be computed, they use more parameters that are all optimised towards the training objective. We hypothesise these models should achieve better performance when used in combination with additional, powerful regularisers (priors) driven by additionally available data (as discussed earlier) than using hand-crafted features that do not improve with such extra data. Furthermore, end-to-end architectures can be used to develop "universal" models that can generalise across multiple different tasks due to their flexibility – a prior on their parameters and therefore the space of input features extracted in each layer can be constructed using the additionally available data using methods such as multi-task learning [Caruana, 1998], self-supervised pre-training [Devlin et al., 2019] or meta-learning [Finn et al., 2017]. On the other hand, models using a fixed set of input features designed for one task tend to perform poorly on other tasks that require different features.

1.2 Aim

This work aims to contribute towards the development of deep learning models that generalise well from only limited amounts of labelled data. To this end,

we investigate approaches that leverage additional data from other sources to regularise DL models and thereby prevent overfitting to the task's training set. Experimental evaluation of our approaches will focus on machine listening applications and music information retrieval in particular, but other application domains are also investigated since our approaches are mostly applicable across different domains.

Models should be flexible in the presence of missing data, i.e., should also be able to learn from incomplete observations, since this effectively increases the size of the available training dataset. In audio source separation for example, DL models are usually trained in a supervised fashion to estimate audio sources from an audio mixture, but this requires multi-track data that yields mixtures together with their sources, which is often limited. However, additional datasets with individual mixtures or source recordings might be available. Integrating these datasets can be achieved when framing the problem as a missing data scenario in which samples from these datasets are only partially observable.

Another approach to obtain "data-driven regularisation" as outlined above is multi-task learning, where a model is also required to perform additional tasks that are assumed to be related to the main task of interest. For example, detecting the presence of musical instruments and recovering their corresponding audio signals from a mixture (separation) are closely related tasks since the target labels of both are correlated – when an instrument labelled as not active at a particular time, it should be near-silent. Therefore, leveraging multiple datasets each featuring annotations for a different tasks by including them into model training is a promising research direction.

Finally, we develop end-to-end neural architectures for processing and generating audio signals in an end-to-end fashion. These architectures can pave the way towards models that are more universally applicable across different audio processing tasks, which enable the use of multi-task learning and other data-driven regularisation methods we investigate in this thesis. In current machine listening approaches, model inputs normally vary between tasks due to task-specific feature engineering – for example, audio source separation is normally performed on a spectrogram representation, but for many audio detection tasks, the spectrogram is often compressed further before being input to a statistical model. We also investigate architectures which can be minimally adapted to work on other sequential data such as text and symbolic music, which could also enable transfer of models between tasks with different modalities (input domains). Since such data is often very high-dimensional (meaning sequences are often very long, such as raw audio), maximising the computational efficiency of such models is another important goal and so is another focus of our work in this direction.

1.3 Thesis structure

We describe the structure of this thesis in the following by briefly describing the content of each main chapter.

Chapter 2 describes required background knowledge and previous work in areas related to this thesis.

Chapter 3 describes novel methods based on generative adversarial networks (GANs) for a missing data scenario, which are applied to problems such as audio source separation and image segmentation.

Chapter 4 introduces approaches which use related tasks as well as unlabelled data as a model prior to improve generalisation capability, in the context of MIR tasks.

Chapter 5 details new CNN models for temporal data that achieve high computational efficiency and good generalisation capability even with very high resolution (= high-dimensional) input.

Chapter 6 concludes the thesis, summarising the work, discussing its limitations and considering prospects for further research.

1.4 Contributions

The novel contributions of this thesis are:

- Chapter 3: We present novel GAN-based methods for missing data and semi-supervised learning scenarios. Our first method in Section 3.2 combines a standard supervised learning objective with a GAN-based unsupervised constraint in the context of audio source separation. Compared to pure supervised training on a multi-track dataset, we can improve performance by leveraging additional solo source recordings.

  In Section 3.3 we further develop the method into a fully adversarial learning framework for this type of missing data scenario, embedding it into the standard GAN framework [Goodfellow et al., 2014]. We apply the method to source separation, image generation and image segmentation and find that including additional incomplete observations improves performance for each task.

- Chapter 4: We present an initial study in Section 4.2 aiming to exploit the shared structure between multiple MIR tasks. Specifically, we train a model to perform both singing voice separation and singing voice detection

at the same time, using a multi-task learning approach. The resulting model performs better at both tasks compared to training separate models on each task.

In our second study in Section 4.3, we apply meta-learning on a set of related tasks to obtain a weight initialisation that leads to better and quicker generalisation when using it for training on new, unseen tasks than a random initialisation. We do this in the context of music information retrieval, where MIR tasks are often related and only small to medium amounts of labelled data exist. We find that results are more mixed in this setting, where performance sometimes drops when the meta-learned weight initialisation is used. However, in some cases performance increases quite substantially, suggesting the presence of exploitable task similarities, but also a strong dependency of our approach on the selection of training and test tasks.

- Chapter 5: In Section 5.2, we present an end-to-end CNN architecture for audio source separation called the "Wave-U-Net" that improves over spectrogram-based separation by taking the audio input phase into account and avoids artifacts occurring when reconstructing spectrogram magnitudes.

  In Section 5.3, we adapt this architecture to be auto-regressive by making convolutions causal to make it applicable to sequence modelling tasks, obtaining the "Seq-U-Net". We compare to two state-of-the-art sequence models, temporal convolutional networks (TCN) [Bai et al., 2018] and Wavenet [van den Oord et al., 2016] on text, symbolic audio and raw audio generation tasks and find it delivers similar performance while using significantly less memory and computation time.

1.5 Associated publications

Portions of the work detailed in this thesis have been presented in national and international scholarly publications, as follows :

- Work described in Section 3.2 was published at the International Conference on Acoustics, Speech, and Signal Processing (ICASSP) [Stoller et al., 2018b].
- Work described in Section 3.3 was published at the International Conference on Learning Representations (ICLR) [Stoller et al., 2020a].

– Work described in Section 4.2 was published at the 14th International Conference on Latent Variable Analysis and Signal Separation (LVA ICA 2018) [Stoller et al., 2018c].
– Work described in Section 5.2 was published at the International Society for Music Information Retrieval Conference (ISMIR) [Stoller et al., 2018d].
– Work described in Section 5.3 was published at the International Joint Conference on Artificial Intelligence and the Pacific Rim International Conference on Artificial Intelligence (IJCAI-PRICAI) [Stoller et al., 2020b].

All work described in this thesis was conducted by the author, including the implementation of experiments, development of code, and writing, with general guidance and feedback given by his supervisors.

Further work not related to the main topic of the PhD was presented in national and international scholarly publications, as follows:

– Adrien Ycart, Daniel Stoller, and Emmanouil Benetos. A comparative study of neural models for polyphonic music sequence transduction. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 470–477, 2019
– Saumitra Mishra, Daniel Stoller, Emmanouil Benetos, Bob L. Sturm, and Simon Dixon. GAN-based generation and automatic selection of explanations for neural networks. In *Safe Machine Learning Workshop at the International Conference on Learning Representations (ICLR)*, 2019
– Bhusan Chettri, Daniel Stoller, Veronica Morfi, Marco A. Martínez Ramírez, Emmanouil Benetos, and Bob L. Sturm. Ensemble models for spoofing detection in automatic speaker verification. In *Proceedings of INTERSPEECH*, pages 1018–1022, 2019
– Igor Vatolkin and Daniel Stoller. Evolutionary multi-objective training set selection of data instances and augmentations for vocal detection. In *8th International Conference on Computational Intelligence in Music, Sound, Art and Design (EvoMUSART)*, pages 201–216, 2019. DOI: 10.1007/978-3-030-16667-0_14
– D. Stoller, V. Akkermans, and S. Dixon. Detection of Cut-Points for Automatic Music Rearrangement. In *2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2018a. DOI: 10.1109/MLSP.2018.8516706

– Daniel Stoller and Simon Dixon. Analysis and classification of phonation modes in singing. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, volume 17, pages 80–86, 2016

Chapter 2

Background

Approaches investigated in this thesis are almost exclusively based on deep learning, and so we will give a brief introduction to deep learning in the following Section 2.1. Then we will describe common applications in machine listening in Section 2.2. A more detailed overview of music information retrieval and audio source separation is given in Sections 2.2.1 and 2.2.2, since many limitations of the current approaches in these fields motivate the work in this thesis.

Our goal is to leverage the capability of deep learning but avoid overfitting in the presence of limited annotations. Sections 2.3, 2.4, and 2.5 thus describe methods to regularise models in a data-driven fashion in various scenarios, and how they were applied so far in the context of machine listening.

End-to-end models have shown great performance in applications where large amounts of data can be used for training [Krizhevsky et al., 2012], because relevant features can be learned based on the data instead of relying on feature engineering. For scenarios with limited annotated data, finding ways to integrate additional data into training of end-to-end models is therefore a more promising approach than using simple models with hand-crafted features. Especially in machine listening applications due to the high sampling rate of audio however, model inputs can be very high-dimensional. This can make training and inference for end-to-end models very computationally expensive. Therefore, we will review the state of the art for handling such sequences in particular in Section 2.6 to identify potential avenues for performance improvements.

2.1 Deep learning

Deep learning (DL) is a subfield of machine learning that primarily deals with "deep neural networks" (DNNs)[1], which process their inputs by applying multiple "layers" that each perform a non-linear transformation of their respective inputs.

DNNs are very successful at approximating a variety of complex functions. This is useful since many problems can be expressed as function estimation problems. For example, image classification requires mapping images to the objects depicted in them. If the images have dimension d (the number of pixels multiplied by number of colour channels) and the system should recognise k different types of objects, it can be modelled as a function $f : \mathbb{R}^d \to [0, 1]^k$ that returns the likelihood of each object occurring in the input image.

2.1.1 Multi-layer perceptron

A multi-layer perceptron (MLP) is a commonly used type of DNN. An input vector $\mathbf{x} \in \mathbb{R}^d$ is processed sequentially by N layers. Every layer $i \in \{1, \ldots, N\}$ consists of k "neurons" that each process the input in a non-linear fashion and output a scalar. More specifically, the set of neurons in a layer can be represented by a $k \times d$ weight matrix $\mathbf{W}_i$ and k-dimensional bias vector $\mathbf{b}_i$, along with an "activation function" f_i that applies a fixed, non-linear transformation separately to each vector element. Subsequent application of all layers yields the overall MLP output

$$\mathbf{o} = f_N(\ldots(f_2(\mathbf{W}_2(f_1(\mathbf{W}_1(\mathbf{x}) + \mathbf{b}_1)) + \mathbf{b}_2)) + \ldots + \mathbf{b}_N). \tag{2.1}$$

Note that the dimensions of the weight matrices and biases need to align so that the output dimensionality of layer i is the same as the required input size for layer $i + 1$.

2.1.2 Convolutional neural network

MLPs feature a lot of trainable parameters – if two consecutive layers have N and M neurons, respectively, the resulting weight matrix for the second layer has dimensions $N \times M$. While the approach of "fully connecting" all neurons of one layer with all neurons of the next one makes MLPs very flexible at solving different tasks, they are also very prone to overfitting and computationally expensive due to the large number of parameters.

But what if dependencies between parts of the input are mostly local and global dependencies only exist on a higher level of abstraction? For example,

[1] It also includes the study of other models such as deep Gaussian processes, but in the context of this thesis DL models and DNNs will be viewed as synonymous.

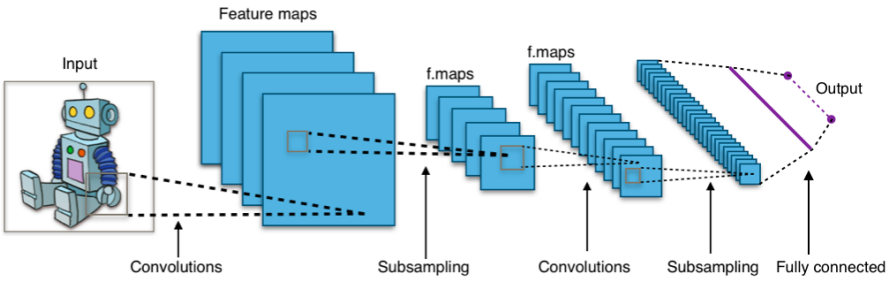


Figure 2.1: Diagram of a convolutional neural network[2]

neighbouring pixels in an image exhibit much stronger correlations than randomly chosen pairs of pixels, and global dependencies arise from the presence of different objects (which are more abstract features). In that case, connecting a neuron in the first layer to all inputs in the next would be inefficient and risk picking up on spurious patterns in the training data, reducing performance.

In convolutional neural networks (CNNs) [Fukushima, 1980] as shown in Figure 2.1, each neuron instead has a "location" relative to the input and is connected only to inputs in its vicinity. Furthermore, sets of neurons share the same weights for their connections, as many patterns should be useful to detect everywhere across the input, which reduces the parameter count further. The above can be implemented by convolving multiple filters with trainable weights across the input, which gives this layer the name "convolutional layer". Its outputs are referred to as "feature maps", as every feature is associated with a specific location in the input.

To compute more abstract features that are more invariant to location, convolutional layers are followed by pooling layers that reduce the spatial resolution of feature maps by downsampling. Convolutional and pooling layers can be repeatedly applied to obtain increasingly abstract features, before finally applying a fully connected layer to obtain the desired output, such as the presence of objects in an image.

2.1.3 Recurrent neural network

Recurrent neural networks (RNNs) are DNNs that apply their transformations repeatedly over a sequence of inputs. Some intermediate outputs computed at each input step (called the hidden state) are fed back to the model itself to be used in the next step, effectively serving as a memory of the past. This makes RNNs very powerful at processing sequences with complex dependencies between their components, since the output at time-step t depends not only on

[2]By Aphex34 - CC BY-SA 4.0, `https://commons.wikimedia.org/w/index.php?curid=45679374`

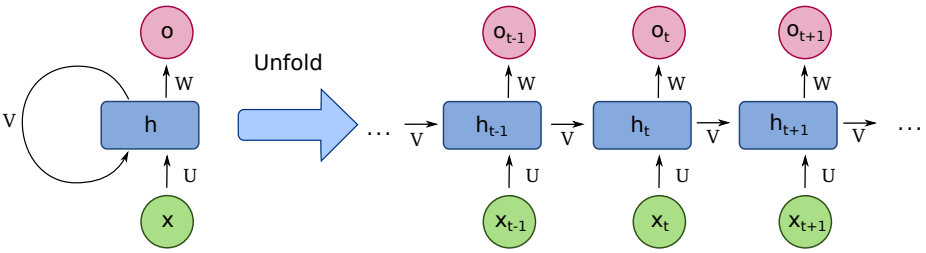


Figure 2.2: Diagram of a recurrent neural network[3]. x is the input vector, h the hidden state, and o the output vector. The model parameters are the input matrix $\mathbf{U}$, the hidden-to-hidden transformation matrix $\mathbf{V}$, and the output matrix $\mathbf{W}$.

the current input x_t, but also implicitly on all previous inputs $x_1, \ldots, x_{t-1}$ due to the dependency on the hidden state h_t, as shown in Figure 2.2. Note that the output of an RNN for a given input sequence is again a sequence – for an input sequence $(x_1, \ldots, x_T)$ of length T, the output is a sequence of output vectors $(o_1, \ldots, o_T)$ that were each computed sequentially by recurrent application of the RNN model.

More specifically, at each input step t, the current input x_t along with the previous hidden state h_{t-1} is used to compute the output o_t and next hidden state h_t as

$$h_t = f_H(\mathbf{V}h_{t-1} + \mathbf{U}x_t) \text{ and} \tag{2.2}$$

$$o_t = f_O(\mathbf{W}h_t), \tag{2.3}$$

where f_H and f_O are suitable activation functions. To compute output o_1, the initial hidden state h_0 is simply set to a constant value. Note that the above formulation depicts a basic RNN and that other variants exist, some of which will be introduced later.

2.1.4 Training deep neural networks

After defining a DNN model, a loss function $L(\theta)$ is defined and subsequently minimised by training the DNN (adjusting its weights θ). The loss function is set depending on the particular problem. In image classification for example, the DNN's output is usually a K-dimensional vector $\mathbf{o}$ indicating for each of the K classes the probability that the input image depicts an object of that class. This vector is compared with the ground truth annotation using a cross-entropy loss that becomes smaller as the model assigns more probability to the correct

[3]By fdeloche - CC BY-SA 4.0, `https://commons.wikimedia.org/w/index.php?curid=60109157`

object class. Furthermore, the loss function is usually continuously differentiable so that gradients of the loss with respect to the weights can be computed using backpropagation [Linnainmaa, 1976], enabling the use of gradient descent for optimisation.

The loss is commonly defined as the expected value of an individual, per-sample objective, with the expectation taken over a dataset of samples. Therefore, stochastic gradient descent (SGD) can be used to approximate the expectation by taking a random set ("batch") of samples from the dataset.

Tackling optimisation issues When training very deep neural networks, training can become ineffective as gradients become progressively smaller during backpropagation, leaving very little information about how to update weights in the front layers ("vanishing gradient problem"). This problem was observed in RNNs [Hochreiter, 1998] as they need to be unrolled in time for training (backpropagation through time, BPTT). To backpropagate the gradient for a loss applied to the last RNN output, it needs to be propagated through N applications of the RNN for an input sequence of length N.

Hochreiter [1998] proposed "Long-Short Term Memory" (LSTM), an adaptation of the standard RNN model that uses additive instead of multiplicative updates to the hidden state to avoid this issue. The Gated Recurrent Unit (GRU) [Cho et al., 2014], is a commonly used model based on the LSTM. It has less parameters than the LSTM due to a simplified architecture while achieving similar performance in a wide variety of tasks [Chung et al., 2014].

For DNNs with many layers, optimisation becomes more difficult in general. The average loss on the *training* set achieved after training converges can increase when using more layers in a model, instead of staying the same or decreasing as would be expected due to the higher number of parameters [He et al., 2015]. He et al. [2015] solve this issue using "residual layers", demonstrating that they enable stable training of very deep CNNs. In a residual layer, an input $\mathbf{x}$ is processed by a neural network layer f such as a convolutional layer, and the result $f(\mathbf{x})$ is added to the input to form the residual layer's output

$$\mathbf{y} = \mathbf{x} + f(\mathbf{x}). \tag{2.4}$$

In this way, adding more layers would generally not increase the final training loss as the additional layers can easily be trained to return $f(\mathbf{x}) = 0$ for all inputs $\mathbf{x}$ by reducing weights to zero, so that layer inputs are simply forwarded to the next layer. Note that the addition requires the dimensionality of input $\mathbf{x}$, neural network layer output $f(\mathbf{x})$ and final output $\mathbf{y}$ to be the same, but this problem can be addressed by linearly projecting $\mathbf{x}$ to the required dimensionality

before performing addition.

Overfitting Due to the large amount of parameters of many DNNs, overfitting is a severe problem in many applications, especially those where training data is scarce. Overfitting occurs when the model adapts to spurious patterns in the training data which are not present in the true data distribution, so that it comparatively under-performs on test data not used during training. Many different methods for model regularisation have been proposed to prevent this, e.g. weight decay, Bayesian neural networks [MacKay, 1995] or dropout [Hinton et al., 2012]. However, not all regularisation methods behave equally – some are more effective as they impose a more informative prior on the function space. In this thesis, we will investigate regularisation techniques that use losses on additional datasets than the one originally used for training, as they can arguably impose strong priors on models. This is especially important given our focus on machine listening applications, where labelled datasets are often small, as we will see in the following Section 2.2.

2.2 Machine listening applications

Machine listening is concerned with automating the processing of audio signals with various types of content, such as human speech, music and environmental sounds in general.

Speech signals are the subject of extensive research efforts, since developing machine listening models for this data enables numerous applications such as automatic subtitling of movies and films or automatic translation of spoken content into other languages. Common tasks in this research field include speech recognition [Deng et al., 2013, Sak et al., 2015, Kim et al., 2016, Graves et al., 2013], text-to-speech generation as well as voice transformation [Ling et al., 2015], and speech separation [Wang and Chen, 2018].

Another important type of audio content is music, which is the focus of the music information retrieval (MIR) field. Music is particularly interesting from a research perspective due to its high degree of complexity, as multiple instruments can express various concepts simultaneously over time by using melody, rhythm and timbre, while interacting with each other to form harmonies. We will give a detailed overview of common MIR tasks in Section 2.2.1.

Apart from human-generated audio data such as speech and music, a large space of environmental sounds exists, whose interpretation can be helpful in many contexts. Identifying the location or scene in which a recording was taken is a widely studied problem [Mesaros et al., 2016], and could aid for example to locate emergency calls. Furthermore, automatically recognising animal activity

from sound recordings can be beneficial for monitoring wildlife [Stowell et al., 2016, Nolasco et al., 2019].

Since the thesis has a special emphasis on MIR problems, we will provide an overview of MIR in the following.

2.2.1 Music information retrieval

Music information retrieval is concerned with extracting meaningful information from music signals, but can have a broader interpretation that also includes manipulating and generating music signals as well. We will provide a brief overview of different types of MIR tasks in the following.

One family of MIR tasks involves identifying where certain musical events occur. Notable examples include the detection of note onsets [Schlüter and Böck, 2014], beats [Böck et al., 2016], drum hits [Vogl et al., 2017] or singing voice activity [Lee et al., 2018]. These tasks are usually framed as an event detection problem, and approaches normally consist of training a model to estimate the likelihood of an event occurring at regular time intervals. To obtain the event predictions, this sequence of probabilities is then decoded using methods such as simple thresholding, peak picking, Hidden Markov Models (HMMs) or Deep Belief Networks (DBNs).

Going beyond simply detecting event onsets, transcription tasks aim to obtain a richer description of the musical signal as it evolves over time. Note transcription [Benetos et al., 2013] is the most prominent example of this task, involving the identification of note onset and offset times as well as their musical pitch. Transcribing the fundamental frequency (F0) [Bittner et al., 2017] of the predominant melody in a music signal is another example. In general, other types of tasks with structured outputs exist, such as music segmentation [Foote and Cooper, 2003], where boundaries between musical segments have to be located and optionally these segments have to be labelled according to their musical function.

Detecting whether music pieces as a whole have certain musical properties is another very active field of research. Prominent examples include categorising music pieces into different musical genres Sturm [2012], detecting the musical key [Knees et al., 2015] and predicting the emotions experienced when listening to a music piece [Aljanaki et al., 2016, Black et al., 2014]. More generally, the music tagging task requires identifying which labels (tags) were assigned to each music piece by listeners [Bogdanov et al., 2019], with tags describing musical genres, which instruments are present in the piece, the language of the lyrics, and many other properties. From a machine learning perspective, these tasks are framed as classification or regression problems. In some cases,

both approaches are a reasonable option. One example for this is music tempo estimation, where the tempo is normally given as the number of musical beats per minute (BPM) [Knees et al., 2015]. In a regression setting, a model would directly estimate the BPM, whereas in a classification setting, the continuous range of possible BPM values is subdivided into a fixed number of N tempo bins and the model is asked to perform N-way classification.

For some tasks, models are required to output whole audio signals as predictions. This is the case of audio source separation, where the input signal is assumed to be a mixture of signals from different sources, which should be recovered when given the mixture signal. In the MIR context, the task of music source separation considers the special case of audio source separation, where sources are different musical instruments [Stöter et al., 2018], for example guitars, drums or vocals. Manipulating music signals in meaningful ways is another emerging research field, such as approaches to remix music pieces [Stoller et al., 2018e] or to transfer between musical styles [Mor et al., 2018].

Music information retrieval can also be concerned with note-sheet representations of music instead of audio recordings. Some tasks described above, e.g. genre detection, can be analogously defined for note-sheet inputs [Corrêa and Rodrigues, 2016]. Generating convincing music pieces is also often approached by training models to compose music using some symbolic representation that is usually similar to a note-sheet, before synthesising the instruments according to the obtained symbolic representation to obtain an audio signal [Ycart and Benetos, 2017].

Many tasks in the MIR domain are related, since the musical properties that need to be extracted for the task are often overlapping or correlated. For example, the instruments that are present in a music piece, which need to be extracted in an instrument detection task, correlate with musical genre, and so solving one task can help with solving another. Some tasks stand in a hierarchical relationship with one another, where solving the higher-level tasks requires solving the lower-level task in the process. For example, if one successfully transcribes all instruments in a music piece, instrument activity detection such as singing voice detection [Lee et al., 2018] is also solved in the process since transcribing correctly requires detecting when instruments are active.

Music source separation is another example for a task with many similarities to other tasks, and challenges posed by this problem motivate many approaches in this thesis. Therefore, we will provide an in-depth review of audio and music source separation in the following.

2.2.2 Audio source separation

Audio source separation is the task of predicting the audio signals of sources present in a given audio signal (also called "mixture" as it is a mixture of the source signals). We will review methods for audio source separation in the following, as it is a very suitable "benchmark task" we will use throughout this thesis to evaluate approaches for dealing with small annotated datasets, due to multiple reasons. Firstly, datasets for this task are often small [Rafii et al., 2017] since creating and distributing multi-track data is difficult. At the same time, solo source recordings and individual mixtures are usually available. These cannot be incorporated into the training set when using a standard supervised learning approach, but when viewed as a special kind of missing data (see Section 2.5), could still provide important prior knowledge to the model enabling better generalisation. Source separation also tends to be related to many other classification tasks especially those requiring the detection of activity of one of the sources – e.g. singing voice separation is very closely related to singing voice detection. Finally, audio source separation is suited as a benchmark task since it is a challenging inverse problem due to its under-determined nature: many estimates of the source signals can possibly yield the same mixture signal, as the number of audio channels of the mixture is typically less than the total number of audio channels of the source (change of dimensionality). Additionally, the mapping of a set of source signals to a mixture (the "mixing process") can even be non-linear in some settings and highly complex. For example, an audio engineer might apply additional audio compression and equalisation to a music piece after combining the source signals (individual instruments) into one audio track.

Methods for audio source separation can be divided into two categories. First, we discuss the *generative* approach featuring a Bayesian view in Section 2.2.2.1, where a generative model of the sources (prior) and of the "mixing process" (likelihood) is constructed. Afterwards, posterior inference of the sources given a mixture is performed using this generative model. The *discriminative*[4] approach is presented in Section 2.2.2.2 and forgoes the construction of a generative model. Instead of viewing the desired distribution as the posterior distribution of such a generative model, it is treated simply as a conditional distribution or function that is estimated directly by an ML model, such as an NN.

[4]Note that we use discriminative loosely in the sense of directly approximating a distribution, not limited to just classification but also including regression

2.2.2.1 Generative approaches

A Bayesian perspective is well suited for source separation if extensive prior knowledge about the sources is available, as it can be explicitly integrated into the model. However, early approaches [Févotte, 2007, Cemgil et al., 2007] often had to make many simplifying assumptions about the data generation process to constrain the generative model such that the difficult problem of posterior inference is tractable. A more recent framework provides various entry points to incorporate available prior knowledge into source models [Ozerov et al., 2012]. However, the resulting complex models do not always scale to large data due to the use of computationally intensive inference algorithms.

For modeling singing voice, a commonly made assumption is a sparse representation in the magnitude spectrogram, while the accompaniment is of low rank and changes more slowly [Huang et al., 2012, Chan et al., 2015, Rafii et al., 2013]. However, as indicated by recent evaluation campaigns [Liutkus et al., 2017], modelling more nuanced relationships (using deep networks) might be beneficial since this assumption only holds to an extent.

Non-negative matrix factorisation (NMF) is often used for separation, and can elegantly incorporate prior knowledge about sources [Sun and Mysore, 2013]. However, NMF is limited in expressivity due to the assumption that spectral content can be factorised independently of time [Ewert et al., 2014], and many spectral basis vectors are needed to represent complex instruments.

Overall, current generative models are subject to various constraints in their structure, sacrificing separation performance to keep inference tractable, or require expert knowledge to set priors.

2.2.2.2 Discriminative approaches

Many deep neural networks have been trained to directly predict sources from mixture input, from feed-forward [Nugraha et al., 2015] to convolutional [Simpson et al., 2015, Miron et al., 2017] and recurrent neural networks [Huang et al., 2014, Luo et al., 2017, Uhlich et al., 2017]. The loss involves comparing the prediction and the correct output for each input, restricting the approach to input-output pairs from multi-track datasets. Although these deep architectures perform well, it is not directly possible to improve them by also learning from individual source recordings, since the prior is not explicitly modelled.

Furthermore, extensive data augmentation is required [Uhlich et al., 2017, Miron et al., 2017] to combat overfitting due to the limited number of multi-track recordings. Randomly mixing source excerpts to generate mixtures is common, but assumes that sources are temporally independent. Since this is not the case in music pieces, correlations between sources cannot be exploited by the

separator.

To alleviate the problem of fixed spectral representations widely used in previous work [Nugraha et al., 2015, Simpson et al., 2015, Miron et al., 2017, Huang et al., 2014, Luo et al., 2017, Uhlich et al., 2017], an adaptive front-end for spectrogram computation was developed [Venkataramani and Smaragdis, 2017] that is trained jointly with the separation network, and which operates on the resulting magnitude spectrogram. Despite comparatively increased performance, the model does not exploit the mixture phase for better source magnitude predictions and also does not output the source phase, so the mixture phase has to be used for source signal reconstruction, both of which limit performance.

To our knowledge, only the TasNet [Luo and Mesgarani, 2017] and MR-CAE [Grais et al., 2018] systems tackle the general problem of audio source separation in the time domain. The TasNet performs a decomposition of the signal into a set of basis signals and weights, and then creates a mask over the weights which are finally used to reconstruct the source signals. The model is shown to work for a speech separation task. However, the work makes conceptual trade-offs to allow for low-latency applications, while we focus on offline application, allowing us to exploit a large amount of contextual information.

The multi-resolution convolutional auto-encoder (MRCAE) [Grais et al., 2018] uses two layers of convolution and transposed convolution each. The authors argue that the different convolutional filter sizes detect audio frequencies with different resolutions, but they work only on one time resolution (that of the input), since the network does not perform any resampling. Since input and output consist of only 1025 audio samples (equivalent to 23 ms), it can only exploit very little context information. Furthermore, at test time, output segments are overlapped using a regular spacing and then combined, which differs from how the network is trained. This mismatch and the small context could hurt performance and also explain why the provided sound examples exhibit many artifacts.

For the purpose of speech enhancement and denoising, the SEGAN [Pascual et al., 2017] was developed, employing a neural network with an encoder and decoder pathway that successively halves and doubles the resolution of feature maps in each layer, respectively, and features skip connections between encoder and decoder layers. However, aliasing artifacts can occur in the generated output when using strided transposed convolutions as shown by [Odena et al., 2016]. Furthermore, the model cannot predict audio samples close to its border output well since it is given no additional input context. It is also not clear if the model's performance can transfer to other and more challenging audio source separation tasks.

The Wavenet [van den Oord et al., 2016] was adapted for speech denois-

ing [Rethage et al., 2017] to have a non-causal conditional input and a parallel output of samples for each prediction. It is based on the repeated application of dilated convolutions with exponentially increasing dilation factors to factor in context information. While this architecture is very parameter-efficient, memory consumption is high since each feature map resulting from a dilated convolution still has the original audio's sampling rate as resolution.

Since overfitting is a prevalent issue when employing DL models for audio source separation and many other machine listening tasks due to the limited availability of annotated training data, we will review approaches to improve generalisation in the following.

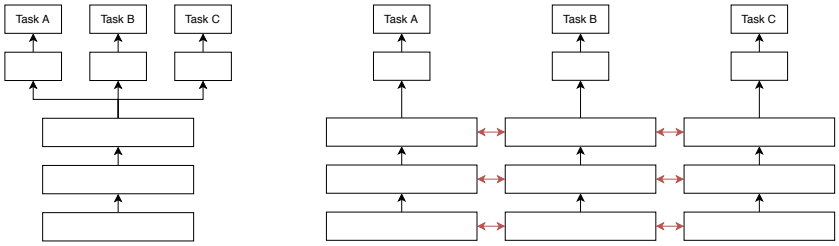
2.3 Multi-task learning

One of the techniques to regularise DL models is to use Multi-task learning (MTL) [Caruana, 1998], which is based on the idea that the features of a model generalise better when they are not only useful for the task at hand, but also for multiple other, related tasks. Besides regularisation, efficiency is another benefit of this approach as a single model can be used for multiple tasks instead of training a separate model for each. The most common ways of realising MTL in DL are called hard and soft parameter sharing, as illustrated in Figure 2.3. In hard parameter sharing, some parameters are shared between all tasks, while others are reserved for individual tasks. Commonly, the initial layers in a DNN are shared, while task-specific layers "branch out" from the final shared layer and process the shared feature set individually as needed, as shown in Figure 2.3a. With soft parameter sharing, each task is tackled with a separate model, but similarity between the parameters in the different models is encouraged by adding some form of weight distance loss (e.g. L2 norm) to the optimisation objective, as illustrated in Figure 2.3b.

Given N tasks, corresponding task losses $L_i, i \in \{1, \ldots, N\}$ and parameters θ_i, the objective optimised in an MTL setting can generally be defined as

$$L(\theta) = \sum_{i=1}^{N} \alpha_i L_i(\theta_i), \tag{2.5}$$

that is, a weighted average of the task losses, where the weights α represent hyper-parameters, and θ is the union of all task-specific parameters. In hard parameter sharing, the parameter sets θ_i would share a common set of parameters. This is not the case in soft parameter sharing, where instead an additional regularising term is added to the objective $L(\cdot)$ based on some distances calculated between pairs of parameter sets θ_i and θ_j where $i \neq j$.



(a) MTL with hard parameter sharing. The first layers are shared between tasks.

(b) MTL with soft parameter sharing. Red arrows indicate parameter constraints.

Figure 2.3: Different MTL approaches in comparison

2.3.1 Application of MTL to MIR

In music information retrieval, many tasks are inter-related in some fashion. Some tasks are arguably in a hierarchical relationship with one another, where solving one task might solve another one in the process – for example, perfectly separating singing voice would make singing voice detection a trivial exercise in measuring the signal energy of the predicted vocal signal, an idea that we will discuss more below and investigate in detail in Section 4.2. Other tasks might exhibit a more "symbiotic" relationship, where certain sub-problems are shared. For example, chords change often at beat positions, so chord detection should benefit from beat detection and beat detection might benefit from observing chord changes – although knowing the solution to one problem does not directly identify the solution to the other, it does provide information due to their correlation.

Applications of MTL to MIR are still few but are becoming increasingly common. However, they are limited to closely related tasks, such as joint beat and downbeat detection [Böck et al., 2016], transcription of onsets, intermediate frames and offsets of notes [Kelz et al., 2019], source separation [Doire and Okubadejo, 2019], and joint multiple-f0, melody, vocal, and bass line estimation [Bittner et al., 2018], and others [Bittner et al., 2018, Chen et al., 2018].

Kim et al. [2019] compare different representation learning strategies for multiple MIR tasks, although they are limited to the setting where an audio excerpt is classified, and so do not consider tasks with time-varying outputs that might also be framed as regression tasks such as beat estimation, transcription and source separation.

MTL in the context of singing voice separation and detection Most approaches to singing voice separation (SVS) train DNNs on multi-track record-

ings in a supervised fashion to estimate the individual sources from a given mixture input [Huang et al., 2014, Luo et al., 2017]. Similarly, recent singing voice detection (SVD) approaches train DNNs as binary classifiers using recordings annotated with changes in singing voice activity [Schlüter, 2016, Lee et al., 2018]. While this approach often leads to considerable improvements over previous methods, it requires suitable multi-track data or annotated recordings, respectively. Unfortunately, publicly available datasets are often rather small on the order of a few hundred tracks. This leads to overfitting and limits overall performance.

Informed source separation aims to circumvent this problem for SVS by providing additional information to the separation model, e.g. the musical score [Ewert and Sandler, 2016]. This way, the problem can be simplified, which often leads to improved results on small, annotated datasets. On the other hand, such approaches can only be employed if suitable side information is indeed available, which is often not the case for musical scores.

A joint separation-classification model [Kong et al., 2017] was proposed for the more general problem of *sound event detection* that employs a separation network whose output mask for each source is summarised with a mean or max operation to detect active sound events. It is designed for weak labels and might be more sensitive to dataset biases when training with different separation and detection datasets due to its simple detection component. Heittola et al. [2011] train with precise activity labels, but separation is used as a front-end for detection instead of performing joint estimation. Therefore, separation cannot be improved using mixtures with only activity labels.

To our knowledge, Chan et al. [2015] provide the only work combining SVS in particular with SVD. Vocal activity labels are used to construct a mask, which forces the corresponding parts of the mixture spectrogram to be modelled individually in a method based on robust principal component analysis (RPCA). For an increase in separation quality however, vocal activity labels are required during prediction. The labels also have to be quite precise as a false negative label would force the vocal estimate to be zero for vocal sections.

Schlüter [2016] focuses solely on SVD, but also shows that the resulting network can be used for detecting the location of the singing voice in the time-frequency domain. This suggests it might be useful to integrate the information contained in the activity labels into separation models to improve their performance. A related method was introduced by Ikemiya et al. [2015]. It produces a rough estimate of the vocals in a first step. After computing the fundamental frequency based on this estimate, the separation result is further refined. These two steps are repeated until convergence.

2.4 Pre-training

Another increasingly common approach to incorporate prior knowledge into a DL model is to use pre-training, where the model is trained on some other task and data before taking the obtained weights and using them as initialisation points when performing the task-specific training. From a Bayesian perspective, this can be viewed as one of many methods to impose a prior on the parameters of an ML model, as the initialisation point defined at the beginning of SGD training can influence how likely the optimisation is to end up in different areas of the parameter space. Note that in contrast to simple feature or representation learning [Bengio et al., 2013], a weight initialisation can include learning a full hierarchy of features along with knowledge about how they are computed, which can be adapted during task-specific training.

We can categorise pre-training methods into three different approaches: unsupervised and supervised pre-training as well as meta-learning of initialisations.

2.4.1 Unsupervised pre-training

The first pre-training approach is called unsupervised pre-training[5] or self-supervised learning. It uses additional unlabelled data together with a hand-crafted loss function designed to prepare the model for the following unseen task. While this allows for potentially large performance gains on the final tasks due to the large amount of unlabelled data that is often available, it is difficult to find a suitable loss function as there are no given target labels to predict.

More recently, pre-training in this manner has achieved great successes [Trinh et al., 2019, Radford et al., 2015], notably in natural language processing (NLP) [Devlin et al., 2019, Peters et al., 2018]. These approaches pre-train large recurrent networks and transformers on an auto-regressive language modelling task, where the next word needs to be predicted given the previous ones. Due to the complex dependencies between words, making good predictions in this context requires learning many important linguistic features such as sentiment, grammatical structure, etc.

There are also approaches to learn useful representations of audio in particular [van den Oord et al., 2019, Chorowski et al., 2019, Schneider et al., 2019], although they are mostly focused solely on speech data. Cramer et al. [2019] construct deep audio embeddings by self-supervised learning of audio-visual correspondence and demonstrate their usefulness when only few labelled data are available, although their results are limited to environmental sound classification tasks and the method requires both video and audio data to be available. For

[5]Note that with "unsupervised pre-training" we do not refer to layer-wise pre-training, as used in very early DL applications [Bengio et al., 2007]

MIR problems, the only existing work employing unsupervised pre-training to our knowledge was conducted by McCallum [2019], which focuses specifically on music segmentation.

2.4.2 Supervised pre-training

The second type of pre-training we identify in this thesis is called supervised pre-training. Similarly to unsupervised pre-training, the model of interest is trained on an additional task before using the obtained weights as initial weights for the actual task training. However, in supervised pre-training, this additional task involves supervised learning, i.e. predicting some kind of label on some dataset, in the hopes that similarities between that label prediction task and the actual task allow for better generalisation. This technique is very commonly used in computer vision, where CNNs are pre-trained on the Imagenet dataset to perform object classification [Girshick et al., 2014, He et al., 2018].

In the audio domain, Kong et al. [2020] pre-train an audio classification model on AudioSet [Gemmeke et al., 2017] before transferring it to six different audio classification tasks. For music specifically, we are only aware of Choi et al. [2018] and van den Oord et al. [2014], who use music tagging as a pre-training task. However, especially for van den Oord et al. [2014], the choice of tasks is constrained to those that are very similar to the pre-training task, e.g. mostly tagging tasks that require identifying high-level attributes of music pieces, as opposed to tasks with time-varying outputs such as music transcription and audio source separation. Additionally, Choi et al. [2018] do not consider fine-tuning the whole DL model after pre-training, but rather use some of the layers to construct a feature set that is then used in other classifiers and compared to basic feature sets, which can be viewed as a more shallow form of pre-training. Furthermore, the performance obtained is often still below that of training the same DL model from scratch.

2.4.3 Meta learning

We identify meta-learning as the third type of pre-training. In meta-learning, "meta-parameters" are optimised so that subsequent training on a task results in good generalisation [Finn et al., 2017, Nichol et al., 2018]. Tasks are assumed to come from a specific task distribution T to enable generalisation to unseen tasks as long as they are also drawn from the same distribution. The meta-objective is to optimise the expected task performance. For example, if we define the weight initialisation at the beginning of task training as a meta-parameter, the meta

objective can be written as

$$\arg\min_{\phi} \mathbb{E}_{\tau \sim T} \left[L_\tau \left(U_\tau^k(\phi) \right) \right], \tag{2.6}$$

where L_τ is the loss for task τ and U_τ^k the operator that updates the model parameters ϕ by performing k steps of gradient descent based optimisation. Note that optimising (2.6) with gradient descent requires backpropagating gradients not only through the final task loss L_τ, but also throughout the whole task training process U_τ^k. Since this can be computationally expensive, a first-order approximation is also proposed by Finn et al. [2017], Nichol et al. [2018] where its gradient is effectively omitted. Meta-learning can be viewed as a type of pre-training if only the weight initialisation used at the beginning of task training is included as a meta-parameter, but it is more general since other components can be meta-learned as well such as the SGD optimiser or the network architecture.

In contrast to the pre-training approaches outlined in Sections 2.4.1 and 2.4.2, where the pre-training objective is assumed to be helpful for the tasks of interest, meta-learning has the advantage of directly optimising for good performance on similar tasks and so can ensure that the pre-training is actually helpful and not detrimental ("negative transfer"). However, it cannot directly make use of unlabelled data in the way unsupervised pre-training is able to, although initial studies begin to tackle this issue [Metz et al., 2019]. Furthermore, in practice it is often limited to few-shot learning settings [Finn et al., 2017, Triantafillou et al., 2018], as each step of meta-optimisation requires training the model on a randomly drawn task or set of tasks with the current meta-parameters. Note that MTL differs from meta-learning in that the usual problem setting of MTL is to perform well on a set of known tasks by solving them jointly, while meta-learning uses these tasks to achieve good performance on a separate set of unseen "test" tasks.

In the context of MIR, we are not aware of any approaches employing meta-learning to enable generalisation across a variety of MIR tasks.

2.5 Tackling missing data

Missing data occurs in many practical scenarios due to various reasons, such as incomplete measurement or drop-out of participants in a study. Therefore, it is important to handle it correctly when building statistical models. In our context, it is desirable to develop models that can be trained from incomplete observations, as it enables the use of additional datasets where certain features are not included. This in turn prevents overfitting to the few samples that were fully annotated for the task.

The missing data problem arises in many applications. Here we will use music source separation, a well established machine listening task, as an example to illustrate the specific kind of missing data problem we consider in this thesis. When separating instruments from music pieces, only few music pieces with the corresponding instrument stems ("multi-track data") are available for training. However, additional solo instrument recordings and individual music pieces are often available. Samples from these additional datasets can also be used for training when framing the task as a special missing data problem, i.e., viewing them as incomplete since some instrument stems or the music mixture is missing.

Importantly, the reason why some values in a dataset are missing needs to be accounted for by the statistical model to avoid introducing bias. For example, the probability that a question A in a survey is answered by a participant might depend on an answer to another question B. If this is ignored and only full observations are used to model the conditional probability of an answer to question A given the answer to question B, the resulting statistical model is biased. For our audio source separation example above, it is important to note that some individual instrument recordings might not be intended to be mixed with other instruments, and so their data distribution is different from instrument recordings taken from the multi-track data.

For the general missing data problem, the reasons for missing values can be categorised into three classes:

Missing completely at random (MCAR) The probability of a value being missing from a sample is completely random, and so does not depend on the value itself or any of the observed values. In this case, incomplete samples are still representative samples in that they come from the same distribution as the complete samples, so no extra care needs to be taken when building statistical models on this data. However, it is rare that MCAR actually applies in practice.

Missing at random (MAR) The probability of a value being missing from a sample is not completely random, but depends only on other *observed* values. Statistical bias can be introduced when not taking account for this.

Missing not at random (MNAR) The probability of a value missing from a sample depends on the value itself.

Note that the reason for missing values is often not known to the researcher, and so one of the above types of missingness is *assumed* to hold for the data in question.

For methods that cannot robustly handle such missing data, the missing values need to be filled in ("imputed") first, for which multiple approaches exist. A simple approach to eliminate incomplete samples is simply to remove them, but this can result in a large loss of information and a too small dataset, as well as dataset bias in case the data is MAR or MNAR. Another method is to fill in each missing value with the mean of the observed values for the respective feature. However, more sophisticated methods for imputation exist, such as multiple imputation methods [Murray, 2018].

A generative adversarial network (GAN) [Goodfellow et al., 2014] is a particularly powerful method for modelling complex probability distributions. However, due to its reliance on large amounts of high-quality data to perform well, it would be desirable to enable its use in various missing data scenarios, so we present GANs and work in this context in the following.

2.5.1 Generative adversarial networks

Generative adversarial networks (GANs) are a powerful approach to modelling complex probability distributions using DL models, which can be employed for various generation and prediction tasks.

To model a probability distribution p_x over $\mathbf{x} \in \mathbb{R}^d$, the standard GAN framework [Goodfellow et al., 2014] introduces a generator model $G_\phi : \mathbb{R}^n \to \mathbb{R}^d$ that maps an n-dimensional input $\mathbf{z} \sim p_z$ to a d-dimensional sample $G_\phi(\mathbf{z})$, resulting in the generator distribution q_x. To train G_ϕ such that q_x approximates the real data density p_x, a discriminator $D_\theta : \mathbb{R}^d \to (0, 1)$ is trained to estimate whether a given sample is real or generated:

$$\hat{\theta} = \arg\max_\theta \ \mathbb{E}_{\mathbf{x}\sim p_x} \log D_\theta(\mathbf{x}) + \mathbb{E}_{\mathbf{x}\sim q_x} \log(1 - D_\theta(\mathbf{x})). \tag{2.7}$$

In the non-parametric limit [Goodfellow et al., 2014], $D_\theta(\mathbf{x})$ approaches $\tilde{D}(\mathbf{x}) := \frac{p_x(\mathbf{x})}{p_x(\mathbf{x})+q_x(\mathbf{x})}$ at every point $\mathbf{x}$. The generator is updated based on the discriminator's estimate of $\tilde{D}(\mathbf{x})$ by minimising $\mathbb{E}_{\mathbf{x}\sim q_x} \log(1 - D_\theta(\mathbf{x}))$ with respect to ϕ. To increase training stability, an alternative loss for G_ϕ is proposed by Goodfellow et al. [2014] that involves minimising

$$L(\theta) = -\mathbb{E}_{\mathbf{z}\sim p_z} \log D_\theta(G_\phi(\mathbf{z})). \tag{2.8}$$

One major problem in training GANs is their instability [Mescheder et al., 2018] – often the generator suddenly collapses to a single output for all noise inputs ("mode collapse"), or the discriminator can "overpower" the generator when it perfectly discriminates real from generated examples, but offers no usable feedback for the generator to improve.

Many alternative formulations for GANs have been developed that use different optimisation objectives or regularise the generator or discriminator networks to improve training stability [Mao et al., 2017, Arjovsky et al., 2017, Gulrajani et al., 2017, Miyato et al., 2018].

In terms of applications, GANs have been successfully used for high-quality image generation [Brock et al., 2019, Karras et al., 2018a,b], image super-resolution [Sønderby et al., 2017] and image-to-image translation [Zhu et al., 2017, Almahairi et al., 2018] and other applications (see Creswell et al. [2018] for an overview).

2.5.2 GANs for missing data scenarios

In the following, we will look at how GANs can be employed in the presence of missing data, either directly as generative models of the joint data distribution or as a data imputation technique.

For semi-supervised classification, GANs were employed to use the discriminator as a regularised classifier [Odena, 2016], to encourage uncertain classifier predictions for generated samples [Springenberg, 2015], and to enforce smoothly changing label predictions around generated points [Miyato et al., 2015].

For conditional generation, "CycleGAN" [Zhu et al., 2017] aims to model conditional distributions $p(A|B)$ and $p(B|A)$ of a given joint distribution $p(A, B)$ using only *unpaired* samples, which are drawn from the two marginal distributions $p(A)$ and $p(B)$. To achieve this, the existence of a one-to-one mapping between the two distributions is assumed and bidirectional generators are used (similarly to Gan et al. [2017]). While CycleGAN can often be trained on more data, since such unpaired samples are often more easily available than *paired* samples drawn directly from the joint $p(A, B)$, also being able to learn from the latter samples would be desirable to more accurately learn the dependency structure. Almahairi et al. [2018] and Tripathy et al. [2018] learn from paired examples with an additional reconstruction-based loss, but use a sum of many different loss terms which have to be balanced by additional hyper-parameters. Additionally, the above methods cannot be applied to generation tasks with missing data or prediction tasks with multiple outputs.

Brakel and Bengio [2017] perform independent component analysis in an adversarial fashion using a discriminator to identify correlations. The separator outputs are enforced to be independent, but the method is not fully adversarial and cannot model arbitrary dependencies. GANs were also used for source separation to improve performance in missing data scenarios [Zhang et al., 2017], but dependencies between sources were ignored.

Pu et al. [2018] use GANs for joint distribution modelling by training a

generator for each possible factorisation of the joint distribution, but this requires $K!$ generators for K marginals. Karaletsos [2016] propose adversarial inference on local factors of a high-dimensional joint distribution and factorise both generator and discriminator based on independence assumptions given by a Bayesian network. Finally, Yoon et al. [2018] randomly mask the inputs to a GAN generator so it learns to impute missing values.

Since we often encounter sequential data in machine listening, such as audio signals, and their high dimensionality makes it challenging to efficiently train DL models that operate on them, we will also review current literature on sequence models in the following Section.

2.6 Sequence models

Since one contribution of our thesis is the development of computationally efficient, end-to-end DNNs for sequences such as audio waveforms, we will provide a review of sequence modeling techniques in the following.

Sequence models are generative models that assign a certain probability to a given sequence $\mathbf{x} = (x_1, \ldots, x_N)$, and are usually trained to maximise the likelihood of the data under the model. Note that this framework also supports conditional generation tasks such as source separation, where source estimates need to be generated for a particular input mixture, when adding the additional condition as input to the model. Since N is normally not constant but rather varies with each sample in the dataset, standard generative models (such as GMMs, flow-based models [Dinh et al., 2016] or GANs) cannot be used directly used as they require the input dimensionality to be fixed. Auto-regressive sequence models avoid this limitation by exploiting the chain rule of probability, which allows factorising the joint probability $p(\mathbf{x})$ as

$$p(\mathbf{x}) = p(x_1) \cdot p(x_2|x_1) \cdot p(x_3|x_1, x_2) \cdot \ldots \cdot p(x_N|x_1, \ldots, x_{N-1}) \tag{2.9}$$

and then using parameters ϕ to model the probability $p_\phi(x_t|x_1, \ldots, x_{t-1})$ of each sequence element x_t given its previous elements. Models can be of limited *order* p, meaning they only consider the previous p elements, thereby estimating $p_\phi(x_t|x_{t-p}, \ldots, x_{t-1})$.

In the following, we give an overview of recent DL-based approaches for sequence modelling, presenting methods employing RNNs (infinite-order models) before discussing methods using CNNs (finite-order models).

2.6.1 RNNs for sequence modelling

Recurrent neural networks (RNNs) are a commonly used approach in deep learning for sequence modelling, including LSTMs and GRUs [Graves et al., 2013, Boulanger-Lewandowski et al., 2012]. In practice, training these models to successfully capture long-term dependencies can be difficult [Bengio et al., 1994] and slow as computation is strictly sequential and cannot be parallelised [Trinh et al., 2018]. Hierarchical multi-scale RNNs [Chung et al., 2016, El Hihi and Bengio, 1995] and the Clockwork RNN [Koutník et al., 2014] model time series on multiple time-scales to enable longer-term dependency modelling, but sequential processing at high-resolution timescales is still computationally expensive. Further work in this direction also includes the DilatedRNN [Chang et al., 2017]. The SampleRNN [Mehri et al., 2016] is a three-layer RNN specifically developed for audio generation. While it also employs a multi-scale approach to an extent, it inherits the disadvantages of RNNs mentioned above, and the "slower" layers have to compute high-level features directly from the raw audio input and forward them to the "faster" layers, which is arguably more difficult than computing them bottom-up using features from the "faster" layers.

2.6.2 CNNs for sequence modelling

Alternative approaches involve CNNs with filters that have increasing dilation factors to cover longer distances between inputs [Kalchbrenner et al., 2016, Campos et al., 2017], of which we highlight TCN [Bai et al., 2018] and Wavenet [van den Oord et al., 2016] for sequence modelling. Due to their depth, these neural models require a large amount of memory and have slow inference as a forward pass is required at each time-step.

The parallel Wavenet [van den Oord et al., 2017] provides fast inference by using a flow-based student network to emulate the outputs of an already trained Wavenet. For long-term dependency modelling in audio, Dieleman et al. [2018] use a complex, multi-stage training with auto-encoding networks to compress the audio before using Wavenets to model the latent state evolution. However, since these approaches involve training a Wavenet, they inherit its computational complexity.

Other approaches have been developed such as FFTNet [Jin et al., 2018], WaveRNN [Kalchbrenner et al., 2018] and MelNet [Vasquez and Lewis, 2019], which do provide large efficiency gains by means of optimisations specific to the audio domain, but at the cost of generality.

Finally, the Transformer network [Vaswani et al., 2017] has shown great potential for sequence-based tasks, but the complexity of its attention mechanism is quadratic in the length of the sequence, preventing its use for long sequences.

Sparse Transformers [Child et al., 2019] restrict the attention modules to a sparse subset of all previous inputs to remedy this, but could still benefit from introducing a multi-scale architecture.

We are unaware of multi-scale approaches evaluated across a variety of sequence modelling problems, but such approaches were used for video segmentation [Shelhamer et al., 2016].

2.7 Discussion

As seen in Sections 2.3 and 2.4, the potential of techniques such as multi-task learning and pre-training has arguably not been explored sufficiently in the context of machine listening and especially MIR. Instead, models are almost always trained to perform a single task at a time, with weights randomly initialised at the beginning of training. Not only is the training time increased compared to approaches that share models between tasks, as models have to start "from scratch" for each task, but also the potential to improve model generalisation by exploiting task similarities remains unused.

Due to research being focused on individual tasks at a time, many proposed NN architectures, regularisation schemes, etc. are only validated on a single MIR task. As a result, it is often uncertain whether the proposed approaches generalise, i.e., yield benefits in other MIR tasks as well.

Furthermore, machine listening models almost always make use of additional feature extraction steps, such as the computation of spectrograms and further derivative features. This often relies on prior knowledge being available about the particular problem, which is limited and can be inaccurate, leading to a glass ceiling in terms of performance. Therefore, it is desirable to research end-to-end models that are enjoying success in other domains such as computer vision [Krizhevsky et al., 2012] and natural language processing [Devlin et al., 2019], and how they can be made flexible enough to support tasks with different input and output requirements. This would in turn further enable exploration of multi-task learning and pre-training techniques as models can be prepared for a larger set of different machine listening tasks.

Due to the lack of feature extraction, however, inputs to end-to-end models can be very high-dimensional. As a result, such models can become very computationally expensive, see e.g. Wavenet [van den Oord et al., 2017]. Improving the computational efficiency of such models is thus another important research direction.

Chapter 3

Generative adversarial networks for missing data scenarios

3.1 Motivation

Real-world datasets often suffer from the problem of missing data, where some observations are incomplete, meaning not all features of interest were observed. Furthermore, datasets from different sources might contain different sets of features that can be observed about the underlying observation. We will focus on one possible missing data setting in particular, where the dataset's feature dimensions can be partitioned into a set of disjoint groups, so that each incomplete observation contains exactly one of these groups of features. These groups are assumed to be missing completely at random (MCAR), which means that the likelihood of a group being missing does not depend on any other data, whether observed or unobserved. In other words, the data distribution from which samples are drawn is the same regardless of which groups are present.

This focus is motivated by a common problem in ASS, where multi-track datasets are required for supervised approaches. While these are difficult to obtain and usually small in size, there are often many additional datasets with isolated source recordings available, which can be viewed as incomplete observations in the missing data scenario explained above – each isolated source recording represents an observation in which all groups of features except for one are missing, and a multi-track recording represents an observation in which all groups are present. By developing methods suited for this scenario, separation models can learn the properties of the sources more precisely than with multi-

track data alone, which in turn allows for higher separation quality as source estimates can be made to sound more realistic. In contrast to manually specifying source priors as commonly done in previous work [Févotte, 2007, Liutkus et al., 2015, Huang et al., 2012], our "source prior" is learnt in a data-driven fashion by providing additional source recordings.

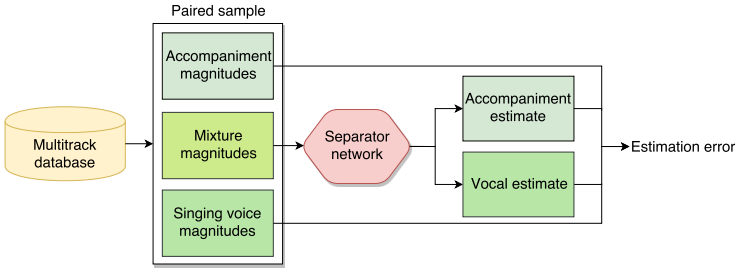
In Section 3.2, we will detail an approach to improving ASS in this setting by combining a standard supervised loss with a novel unsupervised, GAN-based loss. Building on this, Section 3.3 will introduce a generic, fully adversarial framework for handling such missing data along with experiments in a variety of application domains, including not only ASS but also image generation and image segmentation.

3.2 Adversarial semi-supervised audio source separation

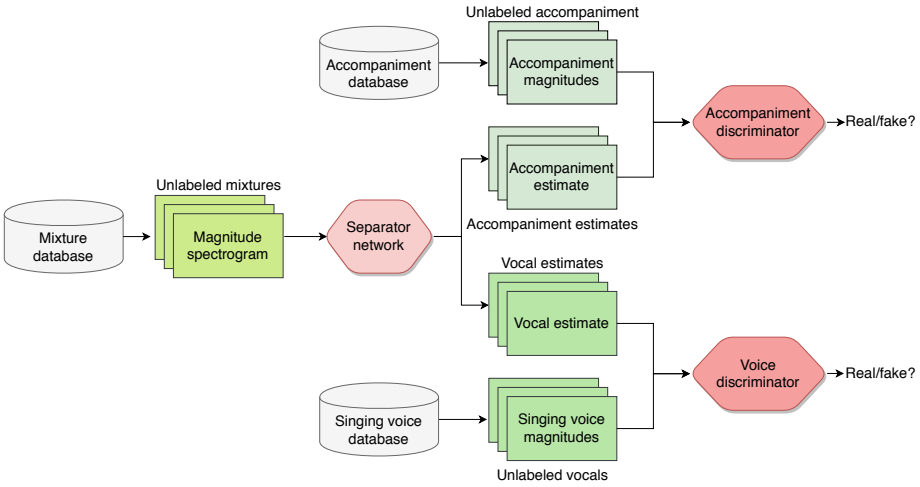
In our first study, we will focus on ASS. Separating instruments from music recordings is challenging as the individual sources are highly correlated in both time and frequency. To approach such a setting, most current methods train deep networks in a supervised manner to predict the source signals for a given mixture input directly. Since the source estimate is compared to the target for each input, as shown in Figure 3.1a, this requires paired input-output samples from multi-track recordings. However, publicly available datasets are rather small, which limits the overall performance. As a result, data augmentation (and other regularisation techniques) are used to combat overfitting [Uhlich et al., 2017, Miron et al., 2017]. However, some of the assumptions made are unrealistic: for example, randomly mixing sources implicitly assumes that sources in a music recording are independent - the reason that music separation is so difficult, however, is exactly due to correlation between instruments. As a result, performance can be limited since source correlations in the test set cannot be learned from the augmented training data.

With a generative approach, we can instead model a prior for each of the sources and how they interact to produce a mixture, the former of which can be learned from solo source recordings. Separation is then an inference problem amounting to finding source estimates that explain a given mixture under the generative model. However, performing the required posterior inference is computationally intensive, so models often have to be heavily simplified [Févotte, 2007], again limiting their performance.

We therefore develop a novel unsupervised objective shown in Figure 3.1b that makes use of the large amount of available unlabelled music tracks as



(a) Standard supervised training



(b) Proposed adversarial training

Figure 3.1: Our proposed system, shown for the example of singing voice separation. In addition to a supervised loss (a), we add an unsupervised loss (b) that drives the separator to produce a more believable output distribution for each source, as assessed by discriminators that try to distinguish real from separator samples.

well as datasets of solo instrument recordings, and combine it with supervised training. This way, we can benefit from the numerical behaviour and training stability of supervised methods while also capturing the variability and richness found in large amounts of unlabelled music and solo instrument recordings. In particular, one discriminator network per source is continually trained to distinguish separator estimates made on full music recordings (unlabelled) from real samples taken from the respective source dataset. The separator aims to output more realistic sources as judged by the discriminators, in addition to minimising the supervised loss on multi-track data.

3.2.1 Proposed framework

Our goal is to separate a mixture m into K sources $\mathbf{s} = (s^1, \ldots, s^K)^T$. Here, each mixture m and source signal s_i represents an excerpt from a magnitude spectrogram - our framework, however, is easily adapted to other input representations such as waveforms. Overall, we assume a multi-track dataset $\mathcal{D}_{\mathrm{m}} = \{(m_1, \mathbf{s}_1), \ldots (m_M, \mathbf{s}_M)\}$ with M input-output samples is available. Furthermore we have access to U unlabelled mixture samples $\mathcal{D}_{\mathrm{u}} = \{m_1^{\mathrm{u}}, \ldots, m_U^{\mathrm{u}}\}$ and a collection $\mathcal{D}_{\mathrm{s}}^k$ of solo recordings for each source k. Let $p(\mathbf{s}, m)$ be the true probability of any source-mixture pair. We assume $\mathcal{D}_{\mathrm{m}}$ is sampled from p, $\mathcal{D}_{\mathrm{u}}$ from the marginal $p_{\mathrm{m}}(m) = \int_{\mathbf{s}} p(\mathbf{s}, m)$, and $\mathcal{D}_{\mathrm{s}}^k$ from the marginal of the k-th source $p_{\mathrm{s}}^k(s^k) = \int_{\{s^1, \ldots, s^K, m\} \setminus \{s^k\}} p(\mathbf{s}, m)$.

In this study, we focus on the case where the separator is deterministic, so that its output probability $q_\phi(\mathbf{s}|m)$ over sources given a mixture with model parameters ϕ can be defined as

$$q_\phi(\mathbf{s}|m) = \delta(f_\phi(m) - \mathbf{s}), \tag{3.1}$$

where $\delta(\cdot)$ is the Dirac delta function, so that $\delta(0)$ yields 1 and otherwise 0.

Our goal is to train the separator function f_ϕ, which can be a deep neural network, with all available data so that it approximates the real posterior $p(\mathbf{s}|m)$. Current approaches usually use the mean squared error between estimates and targets for each input

$$L_{\mathrm{s}} = \frac{1}{M} \sum_{i=1}^{M} ||f_\phi(m_i) - \mathbf{s}_i||_2^2 \tag{3.2}$$

as a loss function on multi-track data. However, this loss function does not include the unlabelled data and is minimised when predicting the posterior mean, which is an unlikely estimate itself and often corresponds to a blurred average of real posterior modes.

We derive an unsupervised loss without these issues. An optimal separator q_ϕ would estimate the real posterior perfectly and thus fulfil $q_\phi(\mathbf{s}|m) = p(\mathbf{s}|m)$ for all possible m. In this case, it follows that the marginal separator output ${}^{\mathrm{out}}q_\phi(\mathbf{s}) = E_{m \sim p_{\mathrm{m}}}\ q_\phi(\mathbf{s}|m)$ would be equal to the true source marginal $p_{\mathrm{s}}(\mathbf{s}) = E_{m \sim p_{\mathrm{m}}}\ p(\mathbf{s}|m)$. If the joint distributions ${}^{\mathrm{out}}q_\phi$ and p_{s} are the same, then so are their source marginals – with ${}^{\mathrm{out}}q_\phi^k(s^k) = \int_{\{s^1, \ldots, s^K\} \setminus \{s^k\}}\ {}^{\mathrm{out}}q_\phi(\mathbf{s})$, this means

$${}^{\mathrm{out}}q_\phi^k = p_{\mathrm{s}}^k,\ \forall\ k = 1, \ldots, K. \tag{3.3}$$

The above distribution equalities are thus necessary, but not sufficient conditions

for an optimal separator.

To approximately fulfil the equalities in (3.3), we can define a divergence $D[{}^{\text{out}}q_\phi^k || p_\text{s}^k]$ with $D[q||p] \geq 0$ and $D[q||p] = 0 \Leftrightarrow q = p$ between the two distributions for each source k to formulate an unsupervised loss we aim to minimise

$$L_\text{u} = \sum_{k=1}^{K} D[{}^{\text{out}}q_\phi^k || p_\text{s}^k]. \tag{3.4}$$

This loss allows using our unlabelled data since we compare K pairs of source distributions instead of individual samples. We approximate p_s^k and ${}^{\text{out}}q_\phi^k$ using batches of samples from the source dataset $\mathcal{D}_\text{s}^k$ and the unlabelled mixture dataset $\mathcal{D}_\text{u}$, respectively.

3.2.1.1 Measurement and choice of divergence

To determine L_u, we need to choose a divergence D and a method to re-estimate it after each separator training step since ${}^{\text{out}}q_\phi$ changes. One possibility is to choose the *Jenson-Shannon* (JS) or *Kullback-Leibler* (KL) divergence as D and use a discriminator network D_{θ_k} for each source k that distinguishes separator from real samples to estimate each divergence - note that the connection between KL and JS divergences and discriminator training is non-trivial, see Goodfellow et al. [2014] for details. With this choice, our unsupervised loss is similar to generative adversarial networks (GANs) [Goodfellow et al., 2014], but we use one discriminator for each source instead of only one, and our "generator" q_ϕ receives mixtures as input instead of random noise.

The original GAN [Goodfellow et al., 2014] is known to be unstable [Arjovsky and Bottou, 2017] since KL and JS divergences can grow to infinity for pairs of distributions that do not overlap. This likely applies to our setting as well, since we use finite sets of samples for ${}^{\text{out}}q_\phi^k$ and p_s^k so they become dirac-like. As a result, the gradients for the separator can vanish or become arbitrarily large near the discriminator's decision boundary, which can destabilise training.

For a stable optimisation, we consider more well-behaved divergences such as the *Wasserstein distance* W_p. As discussed by Gulrajani et al. [2017], the gradient of W_p with respect to the separator output has a bounded norm since a regularising term L_grad is applied on the discriminator networks. Thus we expect separator training to be more stable since the gradient applied to its output does not vanish or explode.

Following the improved Wasserstein GAN algorithm [Gulrajani et al., 2017], we use one discriminator network D_{θ_k} for each source k to approximate the distance $W_p[{}^{\text{out}}q_\phi^k || p_\text{s}^k]$, thereby implementing the unsupervised loss from Equation (3.4). We modify the gradient penalty for the discriminator to be one-sided

since it enabled faster convergence in our experiments:

$$L_{\text{grad}} = E_{x\sim\hat{p}} \max(||\nabla_x D_{\theta_k}(x)||_2 - 1, 0)^2 \tag{3.5}$$

Sampling from $\hat{p}$ involves randomly interpolating between pairs of points sampled from the data and the generator distribution. We use the Wasserstein distance for all following experiments due to instabilities we observed in initial tests using the KL and JS divergences.

3.2.1.2 Additive penalty

For all unlabelled mixtures m_i^{u}, we also aim to ensure that source estimates add up to the mixture, so that $\sum_{k=1}^{K} f_\phi(m_i^{\text{u}})_k \approx m_i^{\text{u}}$. If we know this additive property holds exactly in the true distribution p, and f_ϕ is constrained to only output estimates satisfying this constraint, no additional loss is needed. When using spectrograms, this relationship is only approximate due to phase interference, so we do not constrain the network output $f_\phi(m)$ while minimising the loss

$$L_{\text{add}} = \frac{1}{U}\sum_{i=1}^{U} || \left(\sum_{k=1}^{K} f_\phi(m_i^{\text{u}})_k\right) - m_i^{\text{u}}||_2 \tag{3.6}$$

which is equivalent to maximising $\sum_{i=1}^{U} \log p(m_i^{\text{u}}|f_\theta(m_i^{\text{u}}))$ as likelihood term when assuming $p(m|\mathbf{s})$ is an isotropic Gaussian $\mathcal{N}(m|\sum_{k=1}^{K} s_k; \sigma^2\mathbf{I})$. Since we are considering aggregate priors with L_{u} and a likelihood term $p(m|\mathbf{s})$ with L_{add}, our overall unsupervised training exhibits similarities to variational inference with q_ϕ as inference network that aims to approximate the posterior of the generative model $p(\mathbf{s}, m) = p(m|\mathbf{s}) \prod_{k=1}^{K} p_{\text{s}}^k(s^k)$.

3.2.1.3 Semi-supervised loss

Overall, we minimise the total separator loss $L = L_{\text{s}} + \alpha L_{\text{u}} + \beta L_{\text{add}}$ by stochastic gradient descent. For this, we use a batch of multi-track samples from $\mathcal{D}_{\text{m}}$ to compute L_{s} in the common supervised fashion, as shown in Figure 3.1a. We also use a batch of unlabelled mixtures from $\mathcal{D}_{\text{u}}$ to determine the unsupervised loss L_{u}, as seen in Figure 3.1b. Finally, the additive penalty L_{add} is also computed based on Equation (3.6) if needed – otherwise, the computation is omitted and β is simply set to zero.

After each separator update, we take N_{disc} gradient steps for each discriminator D_{θ_k} to estimate the divergence $W_p[^{\text{out}}q_\phi^k||p_{\text{s}}^k]$ using one shared batch of unlabelled mixtures to generate source estimates and one batch from the respective source dataset $\mathcal{D}_{\text{s}}^k$. The scalars α and β are weights for the loss terms and constitute hyper-parameters.

3.2.2 Singing voice separation experiment

3.2.2.1 Initial considerations

Since the accompaniment in singing voice separation has a very complex distribution, it is harder for the discriminator to estimate the divergence D for the accompaniment than it is for the singing voice. Therefore, we conducted one experiment without the accompaniment discriminator. Note that after removing a divergence term from the unsupervised loss L_{u} it still represents a necessary, albeit weaker, condition for an optimal separator. This is because the optimal separator would still have an unsupervised loss of $L_{\mathrm{u}} = 0$. In other words, the global minima of the original unsupervised loss are retained after removing a divergence term. However, this change creates more global minima with imperfect separators, and in practice we may not find a global minimum at all. This could bias solutions towards favouring vocal over accompaniment quality.

3.2.2.2 Datasets

We use the training partition of the DSD100 [Liutkus et al., 2017] database as our supervised training set $\mathcal{D}_{\mathrm{m}}$. We split the multi-track databases iKala [Chan et al., 2015], MedleyDB [Bittner et al., 2014] and CCMixter [Liutkus et al., 2015] into thirds, and use one third of the tracks from each database to form the unlabelled dataset $\mathcal{D}_{\mathrm{u}}$ and the source datasets $\mathcal{D}_{\mathrm{s}}^{k}$ needed for semi-supervised training. Our validation and test sets are built by taking another third of the tracks from iKala, MedleyDB, and CCMixter, in addition to 25 tracks from the test partition of DSD100. The supervised and unsupervised sets have different sampling biases to enable testing the regularization effect of our semi-supervised approach more directly. We use multi-track data for the unsupervised set despite their known pairing to eliminate dataset bias as a confounding factor, ensuring that differences between the separator output ${}^{\mathrm{out}}q_{\phi}$ and the source dataset distributions stem from the separator. Large databases such as DAMP [Smith, 2013] could be used as unsupervised data, but this study aims at demonstrating a first proof-of-concept and thus does not feature such datasets. Additionally, qualitative differences between such unsupervised data and multi-track stems, for example in recording quality, might limit the potential performance gains of our approach as it assumes the same distribution for supervised and unsupervised data.

3.2.2.3 Experimental setup

Preprocessing The audio input is converted to mono and downsampled to 8 kHz to reduce dimensionality, before the magnitude spectrogram is computed from a 512-point FFT with 50% overlap, and normalized by $x \to \log(1 + x)$. For the unsupervised dataset, we multiply the magnitudes by a factor uniformly drawn from the interval $[0.2, 1.2]$ to make the source discriminators less sensitive to loudness differences. We randomly draw 64 spectrogram excerpts for each batch.

Separator architecture The separator architecture follows the U-Net [Ronneberger et al., 2015, Jansson et al., 2017] closely and uses 3×3 convolutional filters. After a first convolutional layer with 16 filters, 4 downsampling layers perform max-pooling by a factor of 2 followed by a convolution with twice as many filters as the previous layer. The 4 upsampling layers perform transposed convolution with a stride of 2, crop and concatenate the feature map from the respective downsampling layer, before applying another transposed convolution.

The last feature map is concatenated with the mixture input so it can be used as a basis for the output, before K separate 1x1 convolutions are applied, one for each source. After applying ReLU activations, the K feature maps form the K log-normalised source estimates, which are directly input to the discriminators. To generate the final source signals, we use an inverse STFT using the phase from the mixture input. Since the U-Net requires additional context to make predictions at the centre of its input, we use valid convolutions, add temporal context to the input and zero-pad along the frequency axis. We input 158 350-dimensional time frames to retrieve 66 256-dimensional time frames as output.

Discriminator architecture The source discriminators have log-normalised magnitude spectrograms as input and follow the DCGAN architecture [Radford et al., 2015] with Leaky ReLU activations and 32 convolutions in the first layer, with zero-padding in time and frequency. Since the input samples have more frequency bins than time frames, we use two convolutional layers with 4×2 filters and 2×1 stride after the first four layers to detect relationships across frequency bands, before computing 32 dense activations and finally a single linear output.

Training procedure We train the separator and the discriminators on an NVIDIA GTX1080 graphics card using the ADAM optimiser with a learning rate of $5 \cdot 10^{-5}$. Training is stopped if validation performance does not increase after more than six epochs, with 1000 separator update steps in each epoch. Finally, the model with the best validation loss is selected.

Metric	Baseline	V	VA
SDR Inst.	8.09	**8.89**	8.55
SDR V.	6.80	7.28	**7.47**
SIR Inst.	12.03	12.58	**12.67**
SIR V.	13.72	14.00	**14.45**
SAR Inst.	11.27	**12.05**	11.40
SAR V.	8.54	9.00	**9.04**

Table 3.1: Mean performance comparison on the test set (22 instrumental tracks excluded, mono, 8 kHz sampling rate) using the supervised baseline, a vocal discriminator (V) and both vocal and accompaniment discriminators (VA)

A baseline model is trained using only the supervised loss in the log-normalised magnitude space. Then, we train a network with the same architecture using our semi-supervised approach with $\alpha = 0.01$, but without accompaniment discriminator. Finally, we use both discriminators and a lower $\alpha = 0.001$. Each time, we set $\beta = \alpha$. We use low values for α since the losses are not normalised to the same scale by default and in initial tests the loss occasionally plateaued during training, likely due to local minima in the unsupervised loss. Discriminators are trained for $N_{\text{disc}} = 5$ iterations per separator update to re-estimate the respective Wasserstein distance.

3.2.2.4 Evaluation

Quantitative results For evaluation, we calculate the track-wise (normalised) SDR, SIR, and SAR metrics [Vincent et al., 2006], with mono estimates and target signals sampled at 8 kHz. Table 3.1 shows averages over the test set and Table 3.2 over subsets containing only tracks from a specific data source. On the full test set, the purely supervised method ('Baseline') is consistently improved upon across every metric by our method, both with and without an accompaniment discriminator. The baseline method is only better on the DSD100 subset, likely because the supervised set contains only DSD100 training samples, which can be viewed as overfitting. On all other datasets our method yields improvements, especially on the iKala dataset, showing we can train the separator on these samples despite not knowing their input-output pairings. Therefore our unsupervised loss can be a surrogate for the supervised loss enabling learning from unlabelled mixture and source datasets.

Qualitative analysis For an intuition about the discriminator's behaviour, Figure 3.2(a) shows an exemplary vocal estimate from the separator during training, where white denotes high energy. Next to a strong singing voice with vibrato, accompaniment interference is visible as straight, horizontal lines.

	DSD100			MedleyDB		
Metric	Baseline	V	VA	Baseline	V	VA
SDR Inst.	**11.11**	10.75	10.76	9.40	9.60	**9.65**
SDR V.	**3.74**	3.17	3.54	2.48	2.43	**3.00**
SIR Inst.	**14.46**	13.56	13.86	12.18	12.07	**12.74**
SIR V.	**10.03**	9.92	10.49	9.40	9.21	**9.48**
SAR Inst.	14.20	**14.60**	14.10	13.94	**14.23**	13.45
SAR V.	**5.50**	4.84	5.12	4.71	4.69	5.20
	CCMixter			iKala		
Metric	Baseline	V	VA	Baseline	V	VA
SDR Inst.	10.65	**11.09**	10.89	6.34	**7.71**	7.13
SDR V.	3.25	3.52	**3.70**	9.50	10.47	**10.52**
SIR Inst.	15.99	15.49	**16.08**	10.42	**11.79**	11.57
SIR V.	8.39	8.94	**9.35**	16.98	17.44	**17.90**
SAR Inst.	12.84	**13.69**	13.24	9.43	**10.42**	9.70
SAR V.	**6.43**	6.17	6.17	10.81	**11.83**	11.73

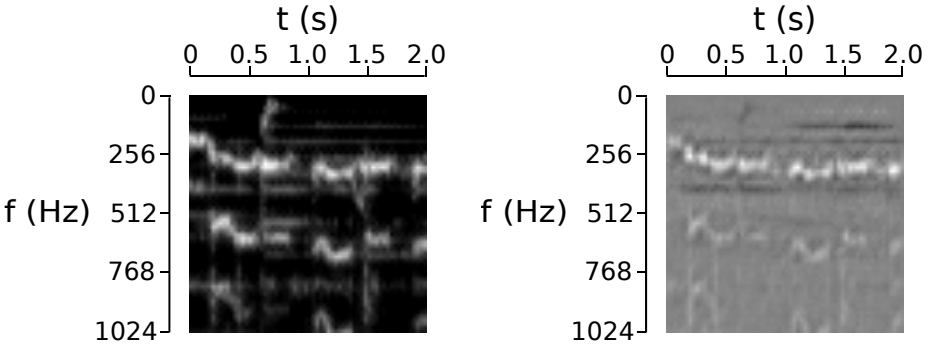
Table 3.2: Mean test set performance comparison on subsets of the test set using the supervised baseline, using a vocal discriminator (V) and using both vocal and accompaniment discriminators (VA)

The discriminator was successfully trained to output large values for real and low values for separator samples: The gradient with respect to the input is positive (shown in white) for vocal parts and negative (shown in black) for the accompaniment artefacts. Therefore the separator is encouraged to attenuate the accompaniment and amplify the voice content to make the voice output more realistic.

3.2.3 Discussion and Conclusion

We presented a semi-supervised framework for ASS. In addition to supervised training on multi-track data, we introduce an unsupervised loss on unlabelled mixtures driving the separator to minimise a divergence between its output distribution and the real source distribution, for each source. The divergence for each source is estimated by its own discriminator continually trained to distinguish real source samples from separator predictions. Our framework is scalable since it can acquire complex source priors from large amounts of unlabelled data while making only few assumptions about the source characteristics.

For singing voice separation, we show an increase in performance compared to purely supervised training. However, performance can also be reduced if the unlabelled data is too scarce or does not come from the same distribution as the test set. Therefore, we used multi-track datasets as our unlabelled data in



(a) Separator estimate x (b) $\nabla_x D(x)$

Figure 3.2: (a) A separator voice estimate x. (b) Gradients of the voice discriminator output with respect to the input x. Only the lower frequency range is shown.

our initial experiment to avoid this confounding factor, but datasets such as DAMP [Smith, 2013] could be included if the dataset bias is slight or can be controlled.

Future work could involve applying our framework to multi-instrument separation due to the highly structured priors for many sources. Our semi-supervised approach also allows training larger separator models that would not generalise sufficiently when trained on multi-track data alone. Finally, discriminator architectures could be adapted to better distinguish separator from real samples and to be less sensitive to the inherent source variability.

Our approach brings noticeable performance improvements in a missing data scenario for ASS. However, it is unsatisfying from a theoretical perspective to combine a supervised loss (such as the MSE loss) with an unsupervised adversarial loss, since it becomes very difficult to characterise the optimal solution for the generator. The supervised and unsupervised losses might "contradict" each other, so that the optimal solutions to the supervised objective do not overlap with the optimal solutions to the unsupervised objective. More specifically, an MSE loss drives the model to output the mean of the posterior distribution, while a GAN objective encourages picking a mode instead.

In the next Section, we will develop a fully adversarial approach to the same problem, and show that it can be applied in other domains such as computer vision.

3.3 Adversarial modelling for missing data

While GANs have become highly effective at synthesising realistic examples even for complex data such as natural images [Radford et al., 2015, Karras et al.,

2018b], they typically rely on large training datasets. These are not available in many cases, especially for prediction tasks such as ASS [Stoller et al., 2018b] or image-to-image translation [Zhu et al., 2017]. Instead, one often encounters many incomplete observations, such as unpaired images in image-to-image translation, or isolated source recordings in source separation. Standard GANs cannot be trained with these observations.

Recent approaches that work with unpaired data can not make use of additional paired data [Zhu et al., 2017] or lead to computational overhead due to additional generators and discriminators that model the inverse of the mapping of interest [Almahairi et al., 2018, Gan et al., 2017]. For training the generator, multiple losses are combined whose interactions are not clear and that do not guarantee that the generator converges to the true distribution. One example for this is our approach described in Section 3.2, which relies on the assumption that the generator (a source separation model in that particular case) is deterministic, i.e. only provides a single estimate for the sources, and does not support probabilistic output densities. In this setting, the use of mean-squared error as supervised loss alongside the GAN-based unsupervised loss means that we cannot characterise the optimal solution for the generator, whereas a fully adversarial framework enables showing that the generator will converge to the true posterior distribution given enough data and training time.

We adapt the standard GAN framework to enable training predictive models with both paired and unpaired data as well as generative models with incomplete observations. To achieve this, we split the discriminator into multiple "marginal" discriminators, each modelling a separate set of dimensions of the input. As this modification on its own would ignore any dependencies between these parts, we incorporate two additional "dependency discriminators", each focusing only on inter-part relationships. We show how the outputs from these marginal and dependency discriminators can be recombined and used to estimate the same density ratios as in the original GAN framework – which enables training any generator network in an unmodified form. In contrast to previous GANs, our approach only requires full observations to train the smaller dependency discriminator and can leverage much larger and simpler datasets to train the marginal discriminators, which enables the generator to model the marginal distributions more accurately. Additionally, prior knowledge about the marginals and dependencies can be incorporated into the architecture of each discriminator. Deriving from first principles, we obtain a consistent adversarial learning framework without the need for extra losses that rely on more assumptions or conflict with the GAN objective.

In our experiments, we apply our approach ("FactorGAN") [1] to two image gen-

[1] Implementation available at `https://github.com/f90/FactorGAN`

eration tasks (Sections 3.3.2.1 and 3.3.2.2), image segmentation (Section 3.3.2.3) and ASS (Section 3.3.2.4), and observe improved performance in missing data scenarios compared to a GAN. For image segmentation, we also compare to the CycleGAN [Zhu et al., 2017], which does not require images to be paired with their segmentation maps. By leveraging both paired and unpaired examples with a unified adversarial objective, we achieve a substantially higher segmentation accuracy even with only 25 paired samples compared to GAN and CycleGAN models.

3.3.1 Method

Firstly, we introduce our method from a missing data perspective below, before extending it to conditional generation (Section 3.3.1.1) and the case of independent outputs (Section 3.3.1.2).

Our method builds on the standard GAN by Goodfellow et al. [2014] that is introduced in Section 2.5.1, using the alternative loss function for the generator G_ϕ. In the following we consider the case that incomplete observations are available in addition to our regular dataset (i.e. simpler yet larger datasets). In particular, we partition the set of d input dimensions of $\mathbf{x}$ into K ($2 \leq K \leq d$) non-overlapping subsets $\mathcal{D}_1, \ldots, \mathcal{D}_K$. For each $i \in \{1, \ldots, K\}$, an incomplete ("marginal") observation $\mathbf{x}^i$ can be drawn from p_x^i, which is obtained from p_x after marginalising out all dimensions not in $\mathcal{D}_i$. Analogously, q_x^i denotes the i-th marginal distribution of the generator G_ϕ. Next, we extend the existing GAN framework such that we can employ the additional incomplete observations. In this context, a main hurdle is that a standard GAN discriminator is trained with samples from the full joint p_x. To eliminate this restriction, we note that $\tilde{D}(\mathbf{x})$ can be mapped to a "joint density ratio" $\frac{p_x(\mathbf{x})}{q_x(\mathbf{x})}$ by applying the bijective function $h : [0, 1) \to \mathbb{R}^+, h(a) = -\frac{a}{a-1}$. For our approach, we exploit that this joint density ratio can be factorised into a product of density ratios:

$$
\begin{aligned}
h(\tilde{D}(\mathbf{x})) = \frac{p_x(\mathbf{x})}{q_x(\mathbf{x})} &= \frac{c_p(\mathbf{x})}{c_q(\mathbf{x})} \prod_{i=1}^{K} \frac{p_x^i(\mathbf{x}^i)}{q_x^i(\mathbf{x}^i)} \text{ with} \\
c_p(\mathbf{x}) &= \frac{p_x(\mathbf{x})}{\prod_{i=1}^{K} p_x^i(\mathbf{x}^i)} \text{ and} \\
c_q(\mathbf{x}) &= \frac{q_x(\mathbf{x})}{\prod_{i=1}^{K} q_x^i(\mathbf{x}^i)}.
\end{aligned}
\tag{3.7}
$$

Each "marginal density ratio" $\frac{p_x^i(\mathbf{x}^i)}{q_x^i(\mathbf{x}^i)}$ captures the generator's output quality for one marginal variable $\mathbf{x}^i$, while the c_p and c_q terms describe the dependency structure between marginal variables in the real and generated distribution,

respectively. Note that our theoretical considerations assume that the densities p_x and q_x are non-zero everywhere. While this might not be fulfilled in practice, our implementation does not directly compute density ratios and instead relies on the same assumptions as Goodfellow et al. [2014]. We can estimate each density ratio independently by training a "sub-discriminator" network, and combine their outputs to estimate $\tilde{D}(\mathbf{x})$, as shown below[2].

Estimating the marginal density ratios: To estimate $\frac{p_x^i(\mathbf{x}^i)}{q_x^i(\mathbf{x}^i)}$ for each $i \in \{1, \dots, K\}$, we train a "marginal discriminator network" $D_{\theta_i} : \mathbb{R}^{|\mathcal{D}_i|} \to (0, 1)$ with parameters θ_i to determine whether a marginal sample $\mathbf{x}^i$ is real or generated following the GAN discriminator loss in Equation (2.7)[3]. This allows making use of the additional incomplete observations. In the non-parametric limit, $D_{\theta_i}(\mathbf{x}^i)$ will approach $\tilde{D}_i(\mathbf{x}^i) := \frac{p_x^i(\mathbf{x}^i)}{p_x^i(\mathbf{x}^i)+q_x^i(\mathbf{x}^i)}$, so that we can use $h(D_{\theta_i}(\mathbf{x}^i))$ as an estimate of $\frac{p_x^i(\mathbf{x}^i)}{q_x^i(\mathbf{x}^i)}$.

Estimation of $c_p(\mathbf{x})$ and $c_q(\mathbf{x})$: Note that c_p and c_q are also density ratios, this time containing a distribution over $\mathbf{x}$ in both the numerator and denominator – the main difference being that in the latter the individual parts $\mathbf{x}^i$ are independent from each other. To approximate the ratio c_p, we can apply the same principles as above and train a "p-dependency discriminator" $D^p_{\theta_p} : \mathbb{R}^d \to (0, 1)$ to distinguish samples from the two distributions, i.e. to discriminate real joint samples from samples where the individual parts are real but were drawn independently of each other (i.e. the individual parts might not originate from the same real joint sample). Again, in the non-parametric limit, its response approaches $\tilde{D}^p(\mathbf{x}) := \frac{p_x(\mathbf{x})}{p_x(\mathbf{x})+\prod_{i=1}^K p_x^i(\mathbf{x}^i)}$ and thus c_p can be approximated via $h \circ D^p_{\theta_p}$. Analogously, the c_q term is estimated with a "q-dependency discriminator" $D^q_{\theta_q}$ – here, we compare joint generator samples with samples where the individual parts were shuffled across several generated samples (to implement the independence assumption).

Joint discriminator sample complexity: In contrast to c_q, where the generator provides an infinite number of samples, estimating c_p without overfitting to the limited number of joint training samples can be challenging. While standard GANs suffer from the same difficulty, our factorisation into specialised sub-units allows for additional opportunities to improve the sample complexity. In particular, we can design the architecture of the p-dependency discriminator

[2]Since the combination of sub-discriminator outputs is used to update the generator, the time complexity of each generator training step grows linearly with the number of marginals K, assuming the time to update each of the K marginal discriminators remains constant.

[3]Samples are drawn from p_x^i and q_x^i instead of p_x and q_x, respectively.

to incorporate prior knowledge about the dependency structure[4].

Combining the discriminators: As the marginal and the p- and q-dependency sub-discriminators provide estimates of their respective density ratios, we can multiply them and apply h^{-1} to obtain the desired ratio $\tilde{D}(\mathbf{x})$, following Equation (3.7). This can be implemented in a simple and stable fashion using a linear combination of pre-activation sub-discriminator outputs followed by a sigmoid. We will prove this in the following.

Definition 3.3.1. Sigmoid discriminator output. Let $D_{\theta_i}(\mathbf{x}^i) := \sigma(d_{\theta_i}(\mathbf{x}^i))$, where $d_{\theta_i} : \mathbb{R}^{|\mathcal{D}_i|} \to \mathbb{R}$ for all $i \in \{1, \dots, K\}$; analogously define $D^p_{\theta_p}(\mathbf{x})$ and $D^q_{\theta_q}(\mathbf{x})$.

Definition 3.3.2. Combined discriminator. Let $D^C(\mathbf{x}) := \sigma(d^p_{\theta_p}(\mathbf{x}) - d^q_{\theta_q}(\mathbf{x}) + \sum_{i=1}^K d_{\theta_i}(\mathbf{x}^i))$ be the output of the combined discriminator that is used for training G_ϕ using Equation 2.8.

Theorem 1. The combined discriminator $D^C(\mathbf{x})$ approximates $\tilde{D}(\mathbf{x})$. Under definitions 3.3.1 and 3.3.2 and assuming optimally trained sub-discriminators (in the non-parametric limit), $D^C(\mathbf{x}) = \tilde{D}(\mathbf{x}) = \frac{p_x(\mathbf{x})}{p_x(\mathbf{x}) + q_x(\mathbf{x})}$.

Proof. Proof of Theorem 1 using Definitions 3.3.1 and 3.3.2:

$$
\begin{aligned}
& D^C(\mathbf{x}) \\
&= \sigma\left(d^p_{\theta_p}(\mathbf{x}) - d^q_{\theta_q}(\mathbf{x}) + \sum_{i=1}^K d_{\theta_i}(\mathbf{x}^i) \right) \\
&= \left(1 + e^{-d^p_{\theta_p}(\mathbf{x})} e^{d^q_{\theta_q}(\mathbf{x})} \prod_{i=1}^K e^{-d_{\theta_i}(\mathbf{x}^i)} \right)^{-1} \\
&= \left(1 + \frac{1 - D^p_{\theta_p}(\mathbf{x})}{D^p_{\theta_p}(\mathbf{x})} \frac{D^q_{\theta_q}(\mathbf{x})}{1 - D^q_{\theta_q}(\mathbf{x})} \prod_{i=1}^K \frac{1 - D_{\theta_i}(\mathbf{x}^i)}{D_{\theta_i}(\mathbf{x}^i)} \right)^{-1} \\
&= \left(1 + \frac{\prod_{i=1}^K p^i_x(\mathbf{x}^i)}{p_x(\mathbf{x})} \frac{q_x(\mathbf{x})}{\prod_{i=1}^K q^i_x(\mathbf{x}^i)} \prod_{i=1}^K \frac{q^i_x(\mathbf{x}^i)}{p^i_x(\mathbf{x}^i)} \right)^{-1} \\
&= \left(1 + \frac{q_x(\mathbf{x})}{p_x(\mathbf{x})} \right)^{-1} \\
&= \frac{p_x(\mathbf{x})}{p_x(\mathbf{x}) + q_x(\mathbf{x})}.
\end{aligned}
\tag{3.8}
$$

□

[4]If only certain features of a marginal variable influence the dependencies, we can limit the input to the p-dependency discriminator to these features instead of the full marginal sample to prevent overfitting.

3.3.1.1 Adaptation to conditional generation

Conditional generation, such as image segmentation or inpainting, can be performed with GANs by using a generator G_ϕ that maps a conditional input $\mathbf{x}^1$ and noise to an output $\mathbf{x}^2$, resulting in an output probability $q_\phi(\mathbf{x}^2|\mathbf{x}^1)$.

When viewing $\mathbf{x}^1$ and $\mathbf{x}^2$ as parts of a joint variable $\mathbf{x} := (\mathbf{x}^1, \mathbf{x}^2)$ with distribution p_x, we can also frame the above task as matching p_x to the joint generator distribution $q_x(\mathbf{x}) := p_x^1(\mathbf{x}^1)q_\phi(\mathbf{x}^2|\mathbf{x}^1)$. Note that this factorisation is obtained by replacing the marginal distribution q_x^1 with the distribution of input examples p_x^1 as they are identical. In a standard conditional GAN, the discriminator is asked to distinguish between joint samples from p_x and q_x, which requires *paired* samples from p_x and is inefficient as the inputs $\mathbf{x}^1$ are the same for both p_x and q_x. In contrast, applying our factorisation principle from Equation (3.7) to $\mathbf{x}^1$ and $\mathbf{x}^2$ (for the special case $K = 2$) yields

$$\frac{p_x(\mathbf{x})}{q_x(\mathbf{x})} = \frac{\frac{p_x(\mathbf{x})}{p_x^1(\mathbf{x}^1)p_x^2(\mathbf{x}^2)}}{\frac{q_x(\mathbf{x})}{q_x^1(\mathbf{x}^1)q_x^2(\mathbf{x}^2)}} \frac{p_x^2(\mathbf{x}^2)}{q_x^2(\mathbf{x}^2)} = \frac{c_p(\mathbf{x})}{c_q(\mathbf{x})} \frac{p_x^2(\mathbf{x}^2)}{q_x^2(\mathbf{x}^2)}, \tag{3.9}$$

suggesting the use of a p- and a q-dependency discriminator to model the input-output relationship, and a marginal discriminator over $\mathbf{x}^2$ that matches aggregate generator predictions from q_x^2 to real output examples from p_x^2. Since the distributions p_x^1 and q_x^1 are equal, $\frac{p_x^1(\mathbf{x}^1)}{q_x^1(\mathbf{x}^1)} = 1$ for all inputs $\mathbf{x}^1$. The term does not appear in (3.9) and we do not need a marginal discriminator for $\mathbf{x}^1$, which increases computational efficiency. The above adaptation to conditional generation can also involve additionally partitioning $\mathbf{x}^2$ into multiple partial observations as shown in Equation 3.7.

3.3.1.2 Adaptation to independent marginals

In case the marginals can be assumed to be completely independent, one can remove the p-dependency discriminator from our framework, since $c_p(\mathbf{x}) = 1$ for all inputs $\mathbf{x}$. This approach can be useful in the conditional setting, when each output is related to the input but their marginals are independent from each other. In this setting, our method is related to adversarial ICA [Brakel and Bengio, 2017]. Note that the q-dependency discriminator still needs to be trained on the full generator outputs if the generator should not introduce unwanted dependencies between the marginals.

3.3.1.3 Further extensions

There are many more ways of partitioning the joint distribution into marginals. We discuss two additional variants (*Hierarchical and auto-regressive FactorGANs*)

of our approach in Section 3.3.3.

3.3.2 Experiments

To validate our method, we compare our FactorGAN with the regular GAN approach, both for unsupervised generation as well as supervised prediction tasks. For the latter, we also compare to the CycleGAN [Zhu et al., 2017] as an unsupervised baseline. To investigate whether FactorGAN makes efficient use of all observations, we vary the proportion of the training samples available for joint sampling (paired), while using the rest to sample from the marginals (unpaired). We train all models using a single NVIDIA GTX 1080 GPU.

Training procedure For stable training, we employ spectral normalisation [Miyato et al., 2018] on each discriminator network to ensure they satisfy a Lipschitz condition. Since the overall output used for training the generator is simply a linear combination of the individual discriminators (see Definition 3.3.1), the generator gradients are also constrained in magnitude accordingly. Unless otherwise noted, we use an Adam optimiser with learning rate 10^{-4} and a batch size of 25 for training all models. We perform two discriminator updates after each generator update.

3.3.2.1 Paired MNIST

Our first experiment will involve "Paired MNIST", a synthetic dataset of low complexity whose dependencies between marginals can be easily controlled. More precisely, we generate a paired version of the original MNIST dataset[5] by creating samples that contain a pair of vertically stacked digit images. With a probability of λ, the lower digit chosen during random generation is the same as the upper one, and different otherwise. For FactorGAN, we model the distributions of upper and lower digits as individual marginal distributions ($K = 2$).

Experimental setup We compare the normal GAN with our FactorGAN, also including a variant without p-dependency discriminator that assumes marginals to be independent ("FactorGAN-no-cp"). We conduct the experiment with $\lambda = 0.1$ and $\lambda = 0.9$ and also vary the amount of training samples available in paired form, while keeping the others as marginal samples only usable by FactorGAN. For both generators and discriminators, we used simple multi-layer perceptrons (Tables 3.3 and 3.4).

To evaluate the quality of generated digits, we adopt the "Frechét Inception Distance" (FID) as metric [Heusel et al., 2017]. It is based on estimating the

[5] `http://yann.lecun.com/exdb/mnist/`

Layer	Input shape	Outputs	Output shape	Activation
FC	50	128	128	ReLU
FC	128	128	128	ReLU
FC	128	1568	$56 \times 28 \times 1$	Sigmoid

Table 3.3: The architecture of our generator on the MNIST dataset. All layers have biases.

Layer	Input shape	Outputs	Output shape	Activation
FC	$W \cdot 28$	128	128	LeakyReLU
FC	128	128	128	LeakyReLU
FC	128	1	1	-

Table 3.4: The architecture of our discriminators on the paired MNIST dataset. $W = 28$ for marginal, $W = 56$ for dependency discriminators.

distance between the distributions of hidden layer activations of a pre-trained Imagenet object detection model for real and fake examples. To adapt the metric to MNIST data, we pre-train a classifier to predict MNIST digits (see Table 3.5) on the training set for 20 epochs, obtaining a test accuracy of 98%. We input the top and bottom digits in each sample separately to the classifier and collect the activations from the last hidden layer (FC1) to compute FIDs for the top and bottom digits, respectively. We use the average of both FIDs to measure the overall output quality of the marginals (a lower value is better).

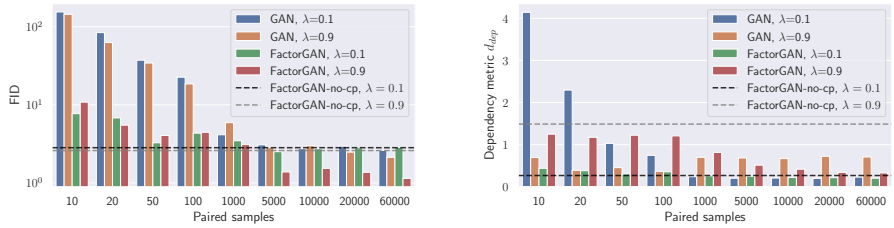
Since the only dependencies in the data are digit correlations controlled by λ, we can evaluate how well FactorGAN models these dependencies. We compute $p_D(D_t, D_b)$ as the probability for a real sample to have digit $D_t \in \{0, \ldots, 9\}$ at the top and digit $D_b \in \{0, \ldots, 9\}$ at the bottom, along with marginal probabilities $p^t_D(D_t)$ and $p^b_D(D_b)$ (and analogously $q_D(D_t, D_b)$ for generated data). Since we do not have ground truth digit labels for the generated samples, we instead use the class predicted by the pre-trained classifier. We encode the dependency as a ratio between a joint and the product of its marginals, where the ratios for real and generated data are ideally the same. Therefore, we take their absolute difference for all digit combinations as evaluation metric (lower is better):

$$d_{\text{dep}} = \frac{1}{100} \sum_{D_t=0}^{9} \sum_{D_b=0}^{9} \left| \frac{p_D(D_t, D_b)}{p^t_D(D_t) p^b_D(D_b)} - \frac{q_D(D_t, D_b)}{q^t_D(D_t) q^b_D(D_b)} \right| . \tag{3.10}$$

Note that the metric computes how well dependencies in the real data are modelled by a generator, but not whether it introduces any additional unwanted

Layer	Input shape	Filter	Stride	Outp.	Output shape	Activation
Conv	$28 \times 28 \times 1$	5×5	1×1	10	$28 \times 28 \times 10$	-
AvgPool	$28 \times 28 \times 10$	2×2	2×2	10	$12 \times 12 \times 10$	LeakyReLU
Conv	$12 \times 12 \times 10$	5×5	1×1	20	$12 \times 12 \times 20$	-
AvgPool	$12 \times 12 \times 20$	2×2	2×2	20	$4 \times 4 \times 20$	LeakyReLU
FC1	320	-	-	50	50	LeakyReLU
FC2	50	-	-	10	10	-

Table 3.5: The architecture of our MNIST classifier. Dropout with probability 0.5 is applied to FC1 outputs.



(a) FID value, averaged over both digits

(b) Dependency metric

Figure 3.3: Performance with different numbers of paired training samples and settings for λ compared between GAN and FactorGAN with and without dependency modelling.

dependencies such as top and bottom digits sharing stroke thickness, and thus presents only a necessary condition for a good generator.

Results The results of our experiment are shown in Figure 3.3. Since FactorGAN-no-cp trains on all samples independently of the number of paired observations, both FID and d_{dep} are constant. As expected, FactorGAN-no-cp delivers good digit quality, and performs well for $\lambda = 0.1$ (as it assumes independence) and badly for $\lambda = 0.9$ with regards to dependency modelling.

FactorGAN outperforms GAN with small numbers of paired samples in terms of FID by exploiting the additional unpaired samples, although this gap closes as both models eventually have access to the same amount of data. FactorGAN also consistently improves in modelling the digit dependencies with an increasing number of paired observations. For $\lambda = 0.1$, this also applies to the normal GAN, although its performance is much worse for small sample sizes as it introduces unwanted digit dependencies. Additionally, its performance appears unstable for $\lambda = 0.9$, where it achieves the best results for a small number of paired examples. Further improvements in this setting could be gained by incorporating prior knowledge about the nature of the dependencies into the p-dependency

Layer	Input shape	Filter	Stride	Outputs	Output shape	Activation
ConvT	$1 \times 1 \times 50$	4×4	1×1	1024	$4 \times 4 \times 1024$	ReLU
ConvT	$4 \times 4 \times 1024$	4×4	2×2	512	$8 \times 8 \times 512$	ReLU
ConvT	$8 \times 8 \times 512$	4×4	2×2	256	$16 \times 16 \times 256$	ReLU
ConvT	$16 \times 16 \times 256$	4×4	2×2	128	$32 \times 32 \times 128$	ReLU
ConvT	$32 \times 32 \times 128$	4×4	2×2	64	$64 \times 64 \times 64$	ReLU
Conv	$64 \times 64 \times 64$	4×4	1×1	C	$64 \times 64 \times C$	Sigmoid

Table 3.6: The architecture of our convolutional generator. "ConvT" represent transposed convolutions. All layers have biases. The number of output channels C depends on the task.

discriminator to increase its sample efficiency, but this is left for future work.

3.3.2.2 Image pair generation

In this section, we use GAN and FactorGAN for generating pairs of images in an unsupervised way to evaluate how well FactorGAN models more complex data distributions.

Datasets For our experiments, we use the "Cityscapes" dataset [Cordts et al., 2016] as well as the "Edges2Shoes" dataset [Isola et al., 2016]. To keep the outputs in a continuous domain, we treat the segmentation maps in the Cityscapes dataset as RGB images, instead of a set of discrete categorical labels. Each input and output image is downsampled to 64×64 pixels as a preprocessing step to reduce computational complexity and to ensure stable GAN training.

Experimental setup We define the distributions of input as well as output images as marginal distributions. Therefore, FactorGAN uses two marginal discriminators and a p- and q-dependency discriminator. All discriminators employ a convolutional architecture shown in Table 3.7 with $W = 6$ and $H = 6$. To control for the impact of discriminator size, we also train a GAN with twice the number of filters in each discriminator layer to match its size with the combined size of the FactorGAN discriminators. The same convolutional generator shown in Table 3.6 is used for GAN and FactorGAN. Each image pair is concatenated along the channel dimension to form one sample, so that $C = 6$ for the Cityscapes and $C = 4$ for the Edges2Shoes dataset (since edge maps are greyscale). We make either 100, 1000, or all training samples available in paired form, to investigate whether FactorGAN can improve upon GAN by exploiting the remaining unpaired samples or match its quality if there are none.

For evaluation, we randomly assign 80% of validation data to a "test-train" and the rest to a "test-test" partition. We train an LSGAN discrimi-

Layer	Input shape	Filter	Stride	Outputs	Output shape
Conv	$2^W \times 2^H \times C$	4×4	2×2	32	$2^{W-1} \times 2^{H-1} \times 32$
Conv	$2^{W-1} \times 2^{H-1} \times 32$	4×4	2×2	64	$2^{W-2} \times 2^{H-2} \times 64$
Conv	$2^{W-2} \times 2^{H-2} \times 64$	4×4	2×2	128	$2^{W-3} \times 2^{H-3} \times 128$
Conv	$2^{W-3} \times 2^{H-3} \times 128$	4×4	2×2	256	$2^{W-4} \times 2^{H-4} \times 256$
Conv	$2^{W-4} \times 2^{H-4} \times 256$	4×4	2×2	512	$2^{W-5} \times 2^{H-5} \times 512$
FC	$2^{W-5} \cdot 2^{H-5} \cdot 512$	-	-	1	1

Table 3.7: The architecture of our convolutional discriminator. All layers except FC have biases and LeakyReLU activations. W, H and C are set for each task so that the dimensions of the input data are matched.

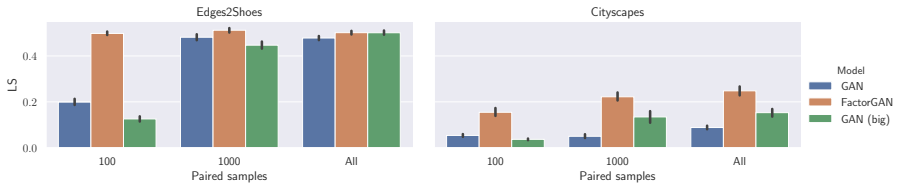


Figure 3.4: GAN and FactorGAN output quality estimated by the LS metric for different datasets and numbers of paired samples. Error bars show 95% confidence intervals.

nator [Mao et al., 2017] with the architecture shown in Table 3.7 (but half the filters in each layer) on the test-train partition for 40 epochs to distinguish real from generated samples, before measuring its loss on the test set. We continuously sample from the generator during training and testing instead of using a fixed set of samples to better approximate the true generator distribution. As evaluation metric, we use the average test loss over 10 training runs, which was shown to correlate with subjective ratings of visual quality [Im et al., 2018] and also with our own quality judgements throughout this study. A larger value indicates better performance, as we use a flipped sign compared to Im et al. [2018]. While the quantitative results appear indicative of output quality, accurate GAN evaluation is still an open problem and so we encourage the reader to judge generated examples given in Section 3.3.5.

Results Our FactorGAN achieves better or similar output quality compared to the GAN baseline in all cases, as seen in Figure 3.4. For the Edges2Shoes dataset, the performance gains are most pronounced for small numbers of paired samples. On the more complex Cityscapes dataset, FactorGAN outperforms GAN by a large margin independent of training set size, even when the discriminators are closely matched in size. This suggests that FactorGAN converges with fewer training iterations for G_ϕ, although the exact cause is unclear and should be

(a) GAN

(b) FactorGAN

Figure 3.5: Examples generated for the Edges2Shoes dataset using 100 paired samples

investigated in future work.

We show some generated examples in Figure 3.5. Due to the small number of available paired samples, we observe a strong mode collapse of the GAN in Figure 3.5a, as the same shoe is generated multiple times and there is not a large variety of different shoes overall. On the other hand, FactorGAN provides high-fidelity, diverse outputs, as shown in Figure 3.5b. Similar observations can be made for the Cityscapes dataset when using 100 paired samples (see Figures 3.22a and 3.22b).

3.3.2.3 Image segmentation

Our approach extends to the case of conditional generation (see Section 3.3.1.1), so we tackle a complex and important image segmentation task on the Cityscapes dataset, where we ask the generator to predict a segmentation map for a city scene (instead of generating both from scratch as in Section 3.3.2.2).

Experimental setup We downsample the scenes and segmentation maps to 128×128 pixels and use a U-Net architecture [Ronneberger et al., 2015] (shown in Table 3.8 with $W = 7$ and $C = 3$) as segmentation model. For FactorGAN, we use one marginal discriminator to match the distribution of real and fake segmentation maps to ensure realistic predictions, which enables training with isolated city scenes and segmentation maps. To ensure the correct predictions for each city scene, p- and q-dependency discriminators learn the input-output relationship using joint samples, both employing the convolutional architecture

Layer	Input (shape)	Outputs	Output shape
DoubleConv1	$2^W \times 128 \times C$	32	$2^W \times 128 \times 32$
MP1	$2^W \times 128 \times 32$	32	$2^{W-1} \times 64 \times 32$
DoubleConv2	$2^{W-1} \times 64 \times 32$	64	$2^{W-1} \times 64 \times 64$
MP2	$2^{W-1} \times 64 \times 64$	64	$2^{W-2} \times 32 \times 64$
DoubleConv3	$2^{W-2} \times 32 \times 64$	64	$2^{W-2} \times 32 \times 128$
MP3	$2^{W-2} \times 32 \times 128$	128	$2^{W-3} \times 16 \times 128$
DoubleConv4	$2^{W-3} \times 16 \times 128$	256	$2^{W-3} \times 16 \times 256$
MP4	$2^{W-3} \times 16 \times 256$	256	$2^{W-4} \times 8 \times 256$
DoubleConv5	$2^{W-4} \times 8 \times 256$	256	$2^{W-4} \times 8 \times 256$
FC	50	$2^{W-4} \cdot 16$	$2^{W-4} \times 8 \times 2$
Concat	DoubleConv5	-	$2^{W-4} \times 8 \times 258$
UpConv	$2^{W-4} \times 8 \times 258$	256	$2^{W-3} \times 16 \times 258$
Concat	DoubleConv4	514	$2^{W-3} \times 16 \times 514$
Conv	$2^{W-3} \times 16 \times 514$	128	$2^{W-3} \times 16 \times 128$
UpConv	$2^{W-3} \times 16 \times 128$	128	$2^{W-2} \times 32 \times 128$
Concat	DoubleConv3	256	$2^{W-2} \times 32 \times 256$
Conv	$2^{W-2} \times 32 \times 256$	64	$2^{W-2} \times 32 \times 64$
UpConv	$2^{W-2} \times 32 \times 64$	64	$2^{W-1} \times 64 \times 64$
Concat	DoubleConv2	128	$2^{W-1} \times 64 \times 128$
Conv	$2^{W-1} \times 64 \times 128$	32	$2^{W-1} \times 64 \times 32$
UpConv	$2^{W-1} \times 64 \times 32$	32	$2^W \times 128 \times 32$
Concat	DoubleConv1	64	$2^W \times 128 \times 64$
Conv	$2^W \times 128 \times 64$	32	$2^W \times 128 \times 32$
Conv	$2^W \times 128 \times 32$	C	$2^W \times 128 \times C$

Table 3.8: The architecture of our U-Net. Height H and number of input channels C depends on the experiment. MP is maxpooling with stride 2. FC has noise as input. UpConv performs transposed convolution with stride 2. DoubleConv is shown in Table 3.9. Concat concatenates the current feature map with one from the downstream path. The final output is computed depending on the task (see text for more details)

shown in Table 3.7. Note that as in Section 3.3.2.2, we output segmentation maps in the RGB space instead of performing classification. In addition to the MSE in the RGB space, we compute the widely used pixel-wise classification accuracy [Cordts et al., 2016] by assigning each output pixel to the class whose colour has the lowest Euclidean distance in RGB space.

Using the same experimental setup (including network architectures), we also implement the CycleGAN [Zhu et al., 2017] as an unsupervised baseline. For the CycleGAN objective, the same GAN losses as shown in (2.7) and (2.8) are used[6].

[6]Code to perform one training iteration and default loss weights were taken from the official codebase at `https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix`

Layer	Input shape	Outputs	Output shape
Conv	$W \times H \times C$	$\frac{C}{2}$	$W \times H \times \frac{C}{2}$
BatchNorm & ReLU	$W \times H \times \frac{C}{2}$	-	$W \times H \times \frac{C}{2}$
Conv	$W \times H \times \frac{C}{2}$	$\frac{C}{2}$	$W \times H \times \frac{C}{2}$
BatchNorm & ReLU	$W \times H \times \frac{C}{2}$	-	$W \times H \times \frac{C}{2}$

Table 3.9: The DoubleConv neural network block used in the U-Net. Conv uses a 3×3 filter size.

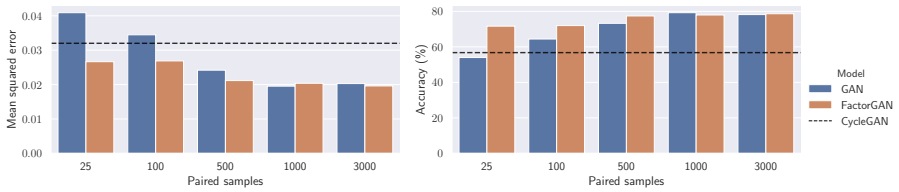
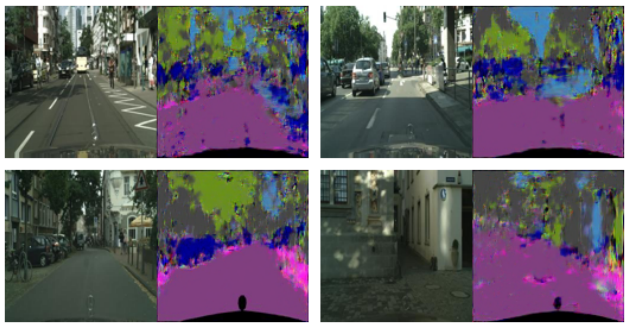


Figure 3.6: MSE (left) and accuracy (right) obtained on the Cityscapes dataset with different numbers of paired training samples for the GAN and FactorGAN

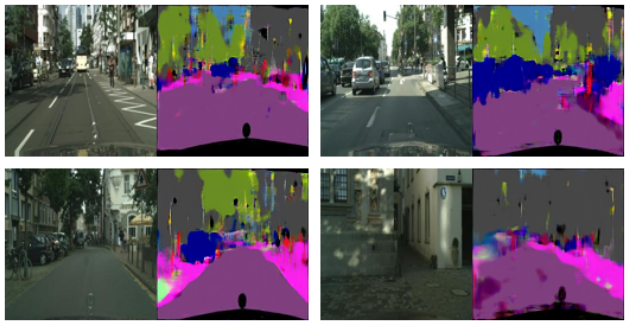
Results The results in Figure 3.6 demonstrate that our approach can exploit additional unpaired samples to deliver better MSE and accuracy than a GAN and less noisy outputs as seen in Figure 3.7. When using only 25 paired samples, FactorGAN reaches 71.6% accuracy, outperforming both GAN and CycleGAN by an absolute 17.7% and 14.9%, respectively. CycleGAN performs better than GAN only in this setting, and increasingly falls behind both GAN and FactorGAN with a growing number of paired samples, likely since GAN and FactorGAN are able to improve their input-output mapping gradually while CycleGAN remains reliant on its cycle consistency assumption. These findings suggest that FactorGAN can efficiently learn the dependency structure from few paired samples with more accuracy than a CycleGAN that is limited by its simplistic cycle consistency assumption.

3.3.2.4 Audio source separation

We apply our method to audio source separation as another conditional generation task. Specifically, we separate music signals into singing voice and accompaniment. For this experiment, our generator G_ϕ takes a music spectrogram $\mathbf{m}$ along with noise $\mathbf{z}$ and maps it to an estimate of the accompaniment and vocal spectra $\mathbf{a}$ and $\mathbf{v}$, implicitly defining an output probability $q_\phi(\mathbf{a}, \mathbf{v}|\mathbf{m})$. We define the joint real and generated distributions that should be matched as $p(\mathbf{m}, \mathbf{a}, \mathbf{v})$ and $q(\mathbf{m}, \mathbf{a}, \mathbf{v}) = q_\phi(\mathbf{a}, \mathbf{v}|\mathbf{m})p(\mathbf{m})$. Since the source signals in our dataset are simply



(a) GAN



(b) FactorGAN

Figure 3.7: Segmentation predictions made on the Cityscapes dataset for the same set of test inputs, compared between models, using 100 paired samples for training

added in the time-domain to produce the mixture, this approximately applies to the spectrogram as well. Therefore, we assume that $p(\mathbf{m}|\mathbf{a},\mathbf{v}) = \delta(\mathbf{m}-\mathbf{a}-\mathbf{v})$, where δ is the Dirac delta distribution. We can constrain our generator G_ϕ to make predictions that always satisfy this condition, thereby taking care of the input-output relationship manually, similarly to Sønderby et al. [2017]. Instead of predicting the sources directly, a mask $\mathbf{b}$ with values in the range $[0,1]$ is computed, and the accompaniment and vocals are estimated as $\mathbf{b}\odot\mathbf{m}$ and $(1-\mathbf{b})\odot\mathbf{m}$, respectively. As a result, $q(\mathbf{m}|\mathbf{a},\mathbf{v}) = p(\mathbf{m}|\mathbf{a},\mathbf{v})$, so we can simplify the joint density ratio to

$$\frac{p(\mathbf{m},\mathbf{a},\mathbf{v})}{q(\mathbf{m},\mathbf{a},\mathbf{v})} = \frac{p(\mathbf{a},\mathbf{v})p(\mathbf{m}|\mathbf{a},\mathbf{v})}{q(\mathbf{a},\mathbf{v})q(\mathbf{m}|\mathbf{a},\mathbf{v})} = \frac{p(\mathbf{a},\mathbf{v})}{q(\mathbf{a},\mathbf{v})} = \frac{c_p(\mathbf{a},\mathbf{v})}{c_q(\mathbf{a},\mathbf{v})}\frac{p(\mathbf{a})}{q(\mathbf{a})}\frac{p(\mathbf{v})}{q(\mathbf{v})}, \tag{3.11}$$

meaning that the discriminator(s) in the GAN and the FactorGAN only require $(\mathbf{a},\mathbf{v})$ pairs, but not the mixture $\mathbf{m}$ as additional input, as the correct input-output relationship is already incorporated into the generator. Furthermore, the last equality suggests a FactorGAN application with one marginal discrim-

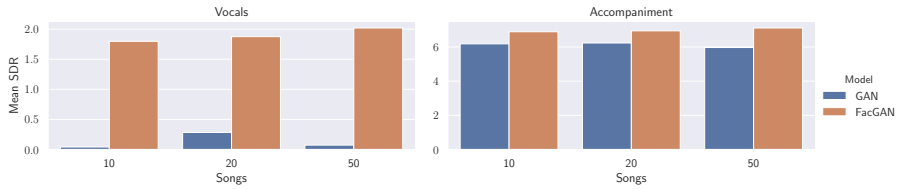


Figure 3.8: GAN and FactorGAN separation performance for different numbers of paired samples

inator for each source along with dependency discriminators to model source dependencies.

Dataset We use MUSDB [Rafii et al., 2017] as multi-track dataset for our experiment, featuring 100 songs for training and 50 songs for testing. Each song is downsampled to 22.05 kHz before spectrogram magnitudes are computed, using an STFT with a 512-sample window and a 256-sample hop[7]. Snippets with 128 timeframes each are created by cropping each song's full spectrogram at regular intervals of 64 timeframes. Thus, the generator only separates snippets $\mathbf{m} \in \mathbb{R}_{\geq 0}^{256 \times 128}$ and outputs predictions of the same shape, however this does not change the derivation presented in Equation (3.11), and longer inputs at test time can be processed by partitioning them into snippets and concatenating the model predictions.

Experimental setup For our generator, we use the U-Net architecture detailed in Table 3.8 with $W = 8$ and $C = 1$. We use the convolutional discriminator described in Table 3.7 with $W = 8$, $H = 7$ and $C = 1$. The source dependency discriminators take two sources as input via concatenation along the channel dimension, so they use $C = 2$.

In each experiment, we vary the number of training songs whose snippets are available for paired training between 10, 20 and 50 and compare between GAN and FactorGAN. The spectrograms predicted on the test set are converted to audio with the inverse STFT by reusing the phase from the mixture, and then evaluated using the signal-to-distortion ratio (SDR), a well-established evaluation metric for source separation [Vincent et al., 2006].

Results Figure 3.8 shows our separation results. Compared to a GAN, the separation performance is significantly higher using FactorGAN. As expected, FactorGAN improves slightly with more paired examples, which is not the case

[7]This results in 257 frequency bins but we discard the bin with the highest frequency to obtain a power of 2 and thus avoid padding issues in our network architectures.

for the GAN – here we find that the vocal output becomes too quiet when increasing the number of songs for training, possibly a sign of mode collapse. Similarly to the results seen in the image pair generation experiments, we suspect that the FactorGAN discriminator might approximate the joint density $\tilde{D}(\mathbf{x})$ more closely than the GAN discriminator due to its use of multiple discriminators, although the reasons for this are not yet understood.

3.3.3 Possible extensions

We can decompose the joint density ratio $\frac{p_x(\mathbf{x})}{q_x(\mathbf{x})}$ in other ways than shown in Equation 3.7. In the following, we discuss two additional possibilities.

3.3.3.1 Hierarchical FactorGAN

The decomposition of the joint density ratio could be applied recursively, splitting the obtained marginals further into "sub-marginals" and their dependencies, which could be repeated multiple times. In addition to training with incomplete observations where only a single part is given, this also allows making use of samples where only sub-parts of these parts are given and is thus more flexible than a single factorisation as used in the standard FactorGAN.

As a demonstration, we split each marginal $\mathbf{x}^i$ further into a group of J_i marginals, $J_i \leq |\mathcal{D}_i|$, and their dependencies, without further recursion for simplicity:

$$\frac{p_x(\mathbf{x})}{q_x(\mathbf{x})} = \frac{c_p(\mathbf{x})}{c_q(\mathbf{x})} \prod_{i=1}^{K} \frac{p_x^i(\mathbf{x}^i)}{q_x^i(\mathbf{x}^i)} = \frac{c_p(\mathbf{x})}{c_q(\mathbf{x})} \left[\prod_{i=1}^{K} \frac{c_p^i(\mathbf{x}^i)}{c_q^i(\mathbf{x}^i)} \left[\prod_{j=1}^{J} \frac{p_x^{i,j}(\mathbf{x}^{i,j})}{q_x^{i,j}(\mathbf{x}^{i,j})} \right] \right]. \tag{3.12}$$

c_p^i and c_q^i are dependency terms analogous to c_p and c_q, but only defined on the marginal variable $\mathbf{x}^i$, whose J "sub-marginals" are denoted by $\mathbf{x}^{i,1}, \ldots, \mathbf{x}^{i,J}$.

Such a hierarchical decomposition might also be beneficial if the data is known to be generated from a hierarchical process. We leave the empirical exploration of this concept to future work.

3.3.3.2 Autoregressive FactorGAN

For a multi-dimensional variable $\mathbf{x} = [\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^T]$ composed of T elements arranged in a sequence, such as time series data, the joint density ratio can also

be decomposed in a causal, auto-regressive fashion:

$$\frac{p_x(\mathbf{x})}{q_x(\mathbf{x})} = \frac{p_x^1(\mathbf{x}^1)}{q_x^1(\mathbf{x}^1)} \prod_{i=2}^{T} \frac{c_p(\mathbf{x}^1, \ldots, \mathbf{x}^i)}{c_q(\mathbf{x}^1, \ldots, \mathbf{x}^i)} \frac{p_x^i(\mathbf{x}^i)}{q_x^i(\mathbf{x}^i)} \tag{3.13}$$

$$= \frac{p_x^1(\mathbf{x}^1)}{q_x^1(\mathbf{x}^1)} \prod_{i=2}^{T} \frac{p_x^i(\mathbf{x}_i|\mathbf{x}_1, \ldots, \mathbf{x}_{i-1})}{q_x^i(\mathbf{x}_i|\mathbf{x}_1, \ldots, \mathbf{x}_{i-1})} \tag{3.14}$$

Note that c_p is defined here as $\frac{p(\mathbf{x})}{p(\mathbf{x}^1,\ldots,\mathbf{x}^{i-1})p(\mathbf{x}^i)}$ (c_q analogously using q_x). Equation (3.13) suggests an auto-regressive version of FactorGAN in which the generator output quality at each time-step i is evaluated using a marginal discriminator that estimates $\frac{p_x^i(\mathbf{x}^i)}{q_x^i(\mathbf{x}^i)}$ combined with dependency discriminators that model the dependency between the current and all past time-steps.

The final product formulation in Equation (3.14) reveals a close similarity to auto-regressive models and suggests a modification of the normal GAN with an auto-regressive discriminator that rates an input at each time-step given the previous ones. Using a derivation analogous to the one shown in (3.8), this implies taking the unnormalised discriminator outputs at each time-step, summing them, and applying a sigmoid non-linearity to obtain the overall estimate of the probability $\tilde{D}(\mathbf{x})$. A similar implementation was used before by Mogren [2016], attempting to stabilise GAN training with recurrent neural networks as discriminators, but for the first time, we provide a rigorous theoretical justification for this practice here.

3.3.4 Discussion and Conclusion

We find that FactorGAN outperforms GAN across all experiments when additional incomplete samples are available, especially when they are abundant in comparison to the number of joint samples. When using only joint observations, FactorGAN should be expected to match the GAN in quality, and it does so quite closely in most of our experiments. Surprisingly, it outperforms GAN in some scenarios such as image segmentation even with matched discriminator sizes – a phenomenon we do not fully understand yet and should be investigated in the future. For image segmentation, FactorGAN substantially improves segmentation accuracy compared to the fully unsupervised CycleGAN model even when only using 25 paired examples, indicating that it can efficiently exploit the pairing information.

Since the p-dependency discriminator does not rely on generator samples that change during training, it could be pre-trained to reduce computation time, but this led to sudden training instabilities in our experiments. We suspect that this is due to a mismatch between training and testing conditions for the

p-dependency discriminator since it is trained on real but evaluated on fake data, and neural networks can yield overly confident predictions outside the support of the training set [Gal and Ghahramani, 2016]. Therefore, we expect classifiers with better uncertainty calibration to alleviate this issue.

We demonstrated how a joint distribution can be factorised into a set of marginals and dependencies, giving rise to the FactorGAN – a GAN in which the discriminator is split into parts that can be independently trained with incomplete observations. For both generation and conditional prediction tasks in multiple domains, we find that FactorGAN outperforms the standard GAN when additional incomplete observations are available. For Cityscapes scene segmentation in particular, FactorGAN achieves a much higher accuracy than the supervised GAN as well as the unsupervised CycleGAN, while requiring only 25 of all examples to be annotated.

3.3.5 Generated examples

We will provide examples generated by our trained models in the following.

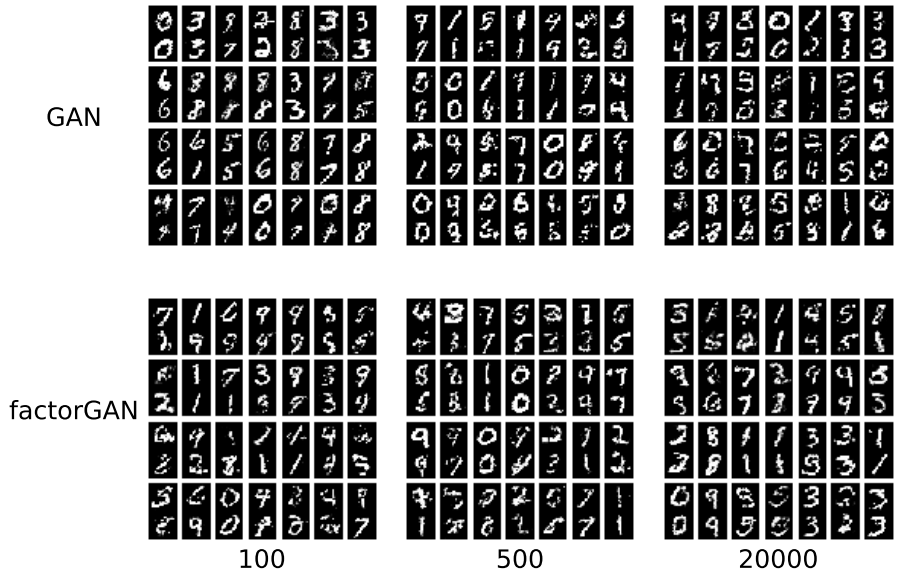


Figure 3.9: Paired MNIST examples generated by GAN and FactorGAN for different number of paired training samples, using $\lambda = 0.9$.

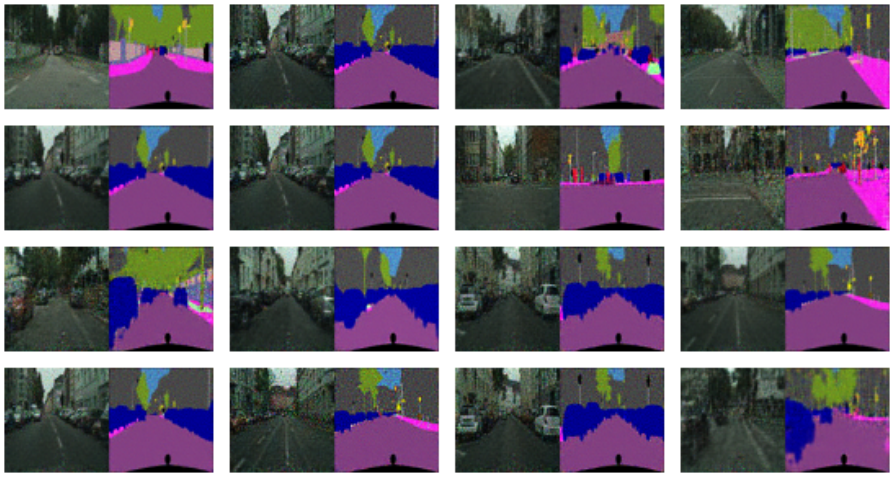


Figure 3.10: GAN generating image pairs for the Cityscapes dataset using 100 paired samples.

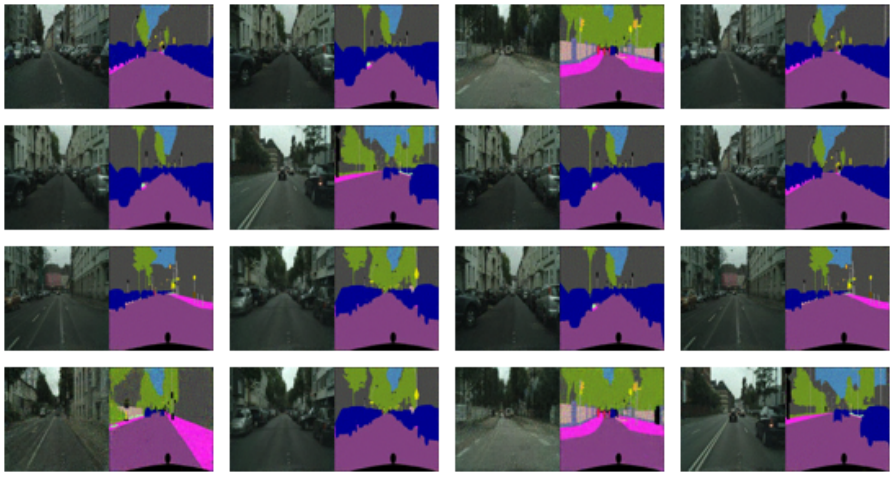


Figure 3.11: GAN (big) generating image pairs for the Cityscapes dataset using 100 paired samples.

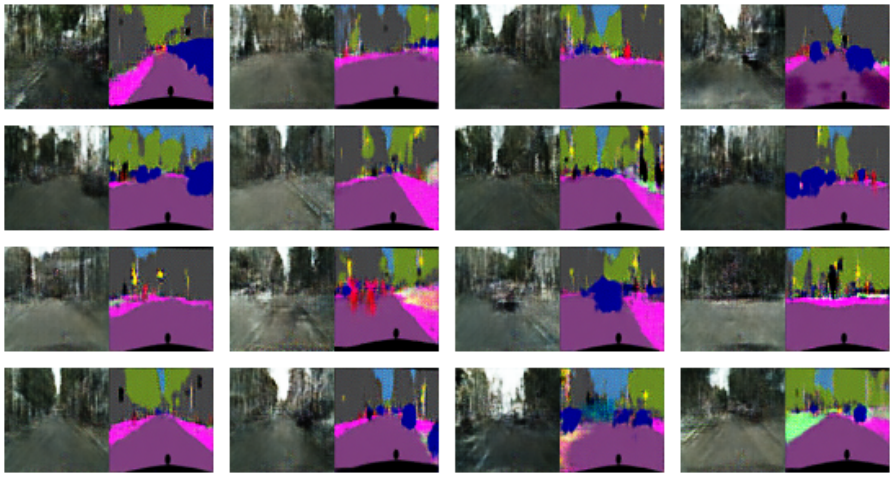


Figure 3.12: FactorGAN generating image pairs for the Cityscapes dataset using 100 paired samples.

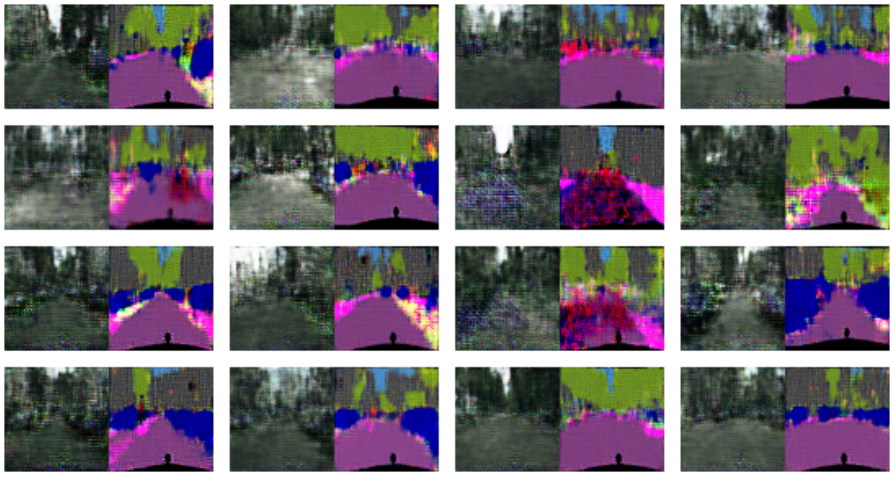


Figure 3.13: GAN generating image pairs for the Cityscapes dataset using 1000 paired samples.

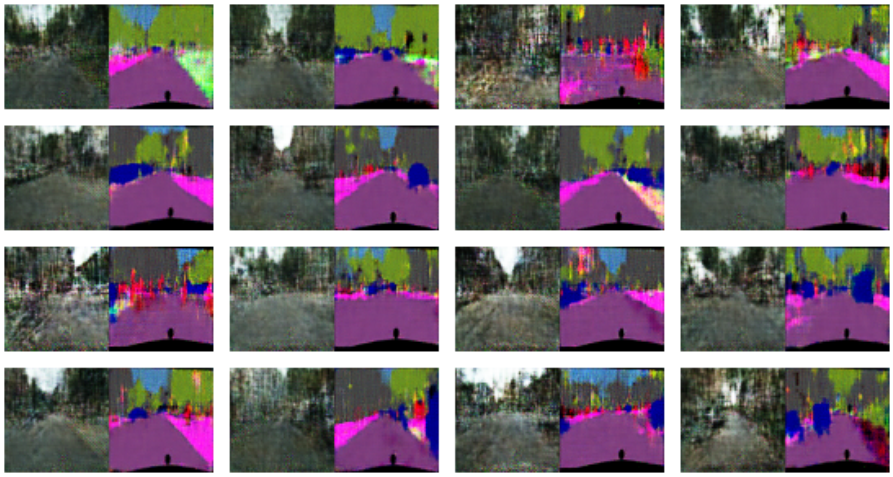


Figure 3.14: GAN (big) generating image pairs for the Cityscapes dataset using 1000 paired samples.

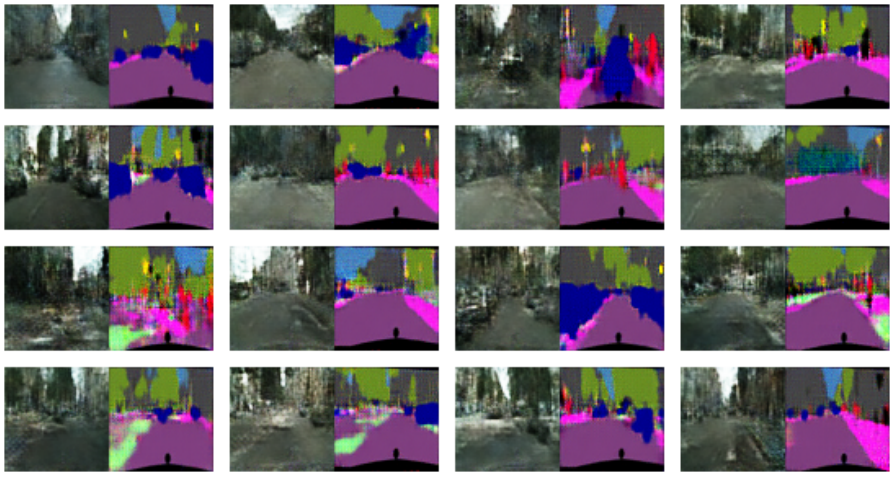


Figure 3.15: FactorGAN generating image pairs for the Cityscapes dataset using 1000 paired samples.

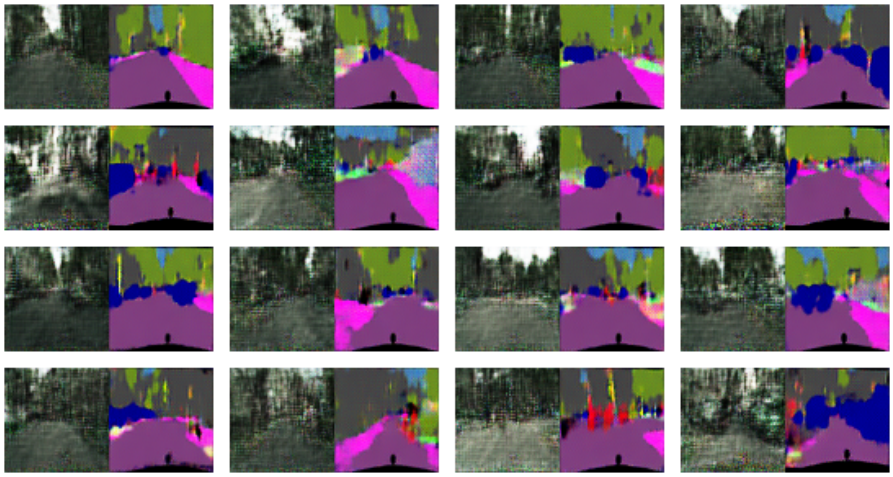


Figure 3.16: GAN generating image pairs using the full Cityscapes dataset.

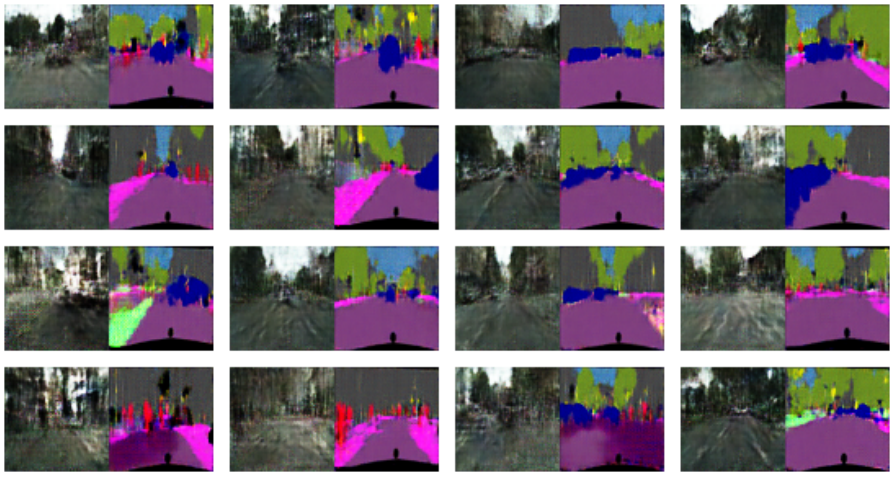


Figure 3.17: GAN (big) generating image pairs using the full Cityscapes dataset.

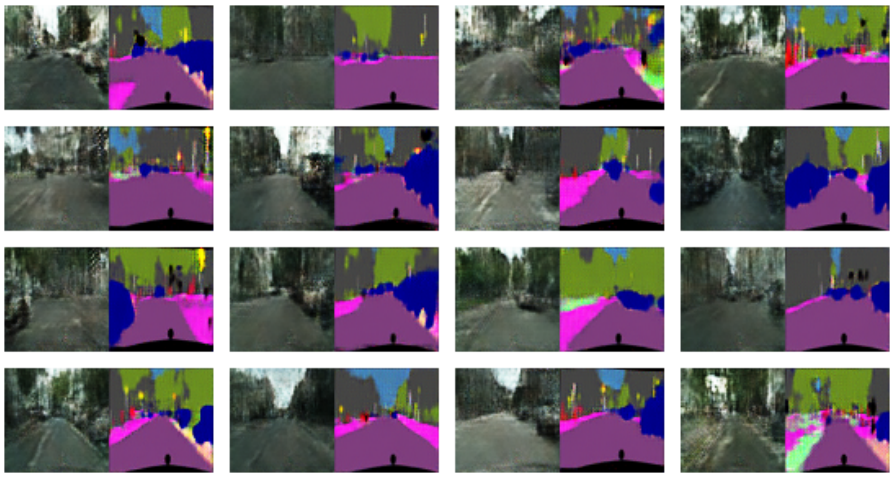


Figure 3.18: FactorGAN generating image pairs using the full Cityscapes dataset.

(a) GAN (b) FactorGAN

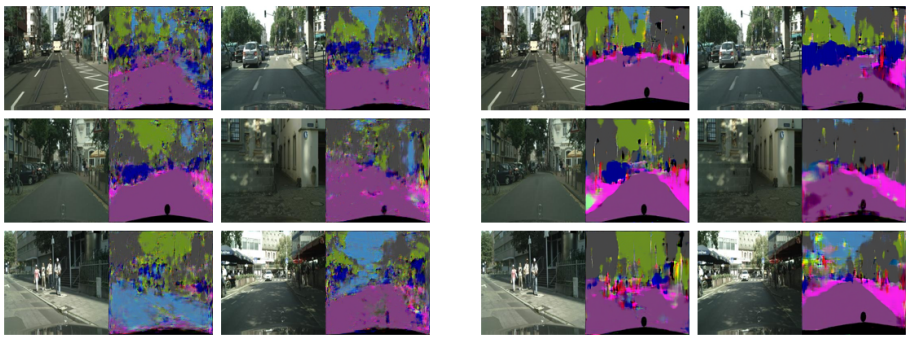
Figure 3.19: Image pairs generated for the Edges2Shoes dataset using 100 paired samples.

(a) GAN (b) FactorGAN

Figure 3.20: Image pairs generated for the Edges2Shoes dataset using 1000 paired samples.

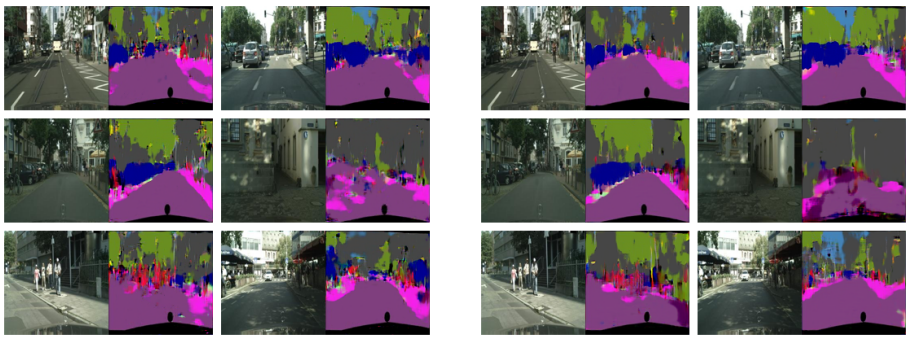
(a) GAN (b) FactorGAN

Figure 3.21: Image pairs generated for the Edges2Shoes dataset using all samples as paired.



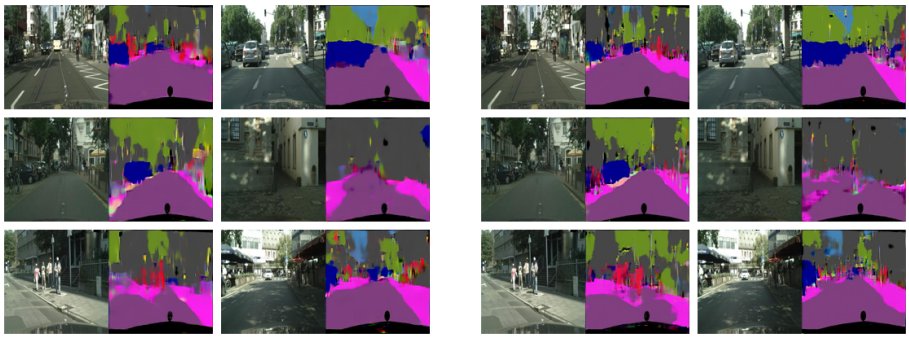
(a) GAN (b) FactorGAN

Figure 3.22: Segmentation predictions made on the Cityscapes dataset for the same set of test inputs, compared between models, using 100 paired samples for training



(a) GAN (b) FactorGAN

Figure 3.23: Segmentation predictions made on the Cityscapes dataset for the same set of test inputs, compared between models, using 1000 paired samples for training



(a) GAN (b) FactorGAN

Figure 3.24: Segmentation predictions made on the Cityscapes dataset for the same set of test inputs, compared between models, using all paired samples for training

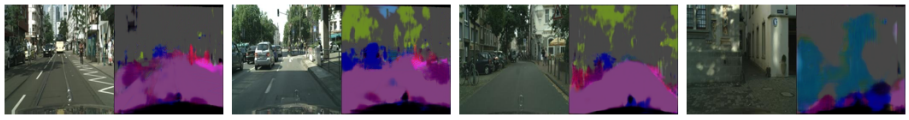


Figure 3.25: CycleGAN generating image pairs for the Cityscapes dataset without any paired samples.

3.4 Conclusion

In the previous Sections 3.2 and 3.3, we presented approaches to integrate additional datasets if they contain a subset of the features contained in the dataset fully annotated for the task, to increase generalisation. In particular, we presented an approach to source separation in Section 3.2 that combines a supervised and unsupervised loss in a simple fashion. Expanding on this concept, we presented a general, fully adversarial framework for handling our missing data scenario. It retains the theoretical guarantees of the original GAN [Goodfellow et al., 2014], so that the generator can be trained in the usual fashion and can be shown to converge to the desired probability distribution. We applied the framework to a variety of problems including image generation, image segmentation and ASS.

In these approaches, we assume the additional data contains some subset of features as the task dataset. But what if the annotations of each dataset concern different aspects of the input data, i.e., they are intended to enabling solving different tasks? In this case, we can not apply the above approaches since the dataset's features are different. Therefore, we will investigate multi-task and meta-learning in the following Section, so that parts of models can be shared between multiple tasks, which can increase computational efficiency but also generalisation. We will focus our experiments on MIR tasks in particular, as they often share considerable overlap between tasks.

Chapter 4

Learned priors for MIR

4.1 Motivation

Tasks in music information retrieval, such as singing voice separation, singing voice detection, segmentation or fundamental frequency estimation, are usually tackled separately [Stöter et al., 2018, Lee et al., 2018, Grill and Schlüter, 2015, Bittner et al., 2017] – models are trained for each task on a task-specific dataset. This approach allows the intuitive design of problem-specific priors when working with traditional ML models, such as hand-crafting particular features. However, deep learning methods that use no or little feature engineering have shown impressive performance on problems for which large-scale annotated training data is available [Krizhevsky et al., 2012], and so MIR researchers have recently explored DL for MIR. While this has resulted in performance improvements compared to traditional methods [Schlüter and Böck, 2014, Sigtia et al., 2015, Grill and Schlüter, 2015], a performance ceiling was quickly reached due to the lack of available training data for many MIR tasks.

Annotating large-scale datasets for each MIR task would be a costly process. Therefore, our approach is to use multi-task learning as well as pre-training to incorporate priors into the model in a data-driven fashion so that it generalises better and more quickly. In multi-task learning, this is achieved by training the model to perform a set of different tasks at the same time, usually with some parts of the model being trained specifically for each task. As a result, model weights shared between the tasks are regularised as they need to provide features useful for multiple tasks. In pre-training, we first train the model on a different set of (labelled or unlabelled) data using a specific loss function. The model weights obtained from this pre-training stage are then used as initialisation points for training on each task. The goal is to initialise the DL model in such a way that it already starts with robust, well generalisable features that can be directly

used at the beginning of task-specific training so that only few training iterations on a small amount of labelled data are required to achieve good performance.

In the next Section 4.2, we will describe an initial study aiming to exploit the similarities between MIR tasks using the specific case of singing voice separation and singing voice detection. Specifically, we will train a model in a multi-task fashion to perform both tasks at the same time, with the goal of improving performance on both tasks. Our second study in Section 4.3 will then turn to meta-learning as a more advanced form of multi-task learning, where models are prepared using multiple training tasks in a way that they perform well on unseen tasks with only little adaptation. Furthermore, we extend our experiments to include ten MIR tasks, aiming to capture more generally useful musical domain knowledge in the resulting models as step towards a "universal" MIR model.

4.2 Joint singing voice separation and detection

Overall, vocal detection and separation are usually tackled as separate tasks despite their commonalities. Thus, a main goal in this section is to explore how such information can be exploited in training audio-only models that can jointly detect and separate vocals. First, we use a simple approach for diversifying the training dataset for a singing voice separation (SVS) model, and observe that its implicit assumption that all data sources are from the same distribution is violated due to a bias specific to each dataset. Using a multi-task learning (MTL) approach, we then propose a model shown in Figure 4.1 that performs SVS and SVD at the same time and can better account for such biases. The model can be trained on multi-track recordings in combination with mixtures with vocal activity labels, and yields predictions on completely unlabelled mixtures. By allowing the model to exploit correlations between the vocal activity labels and the source signals, performance is improved for both tasks compared to baseline models trained with single-task learning (STL). While the overall improvement remained at a rather low level, we found the effect to be quite consistent – despite the small size of the datasets involved and their respective biases. We also found that the most commonly used evaluation metric [Vincent et al., 2006] is flawed in the sense that improvements on non-vocal sections are not captured, and propose a simple ad-hoc solution. As an additional contribution, we discuss the dataset biases we observed in some detail. Overall, based on these findings, we hypothesise that the joint prediction of source estimates along with side information such as musical scores in a multi-task setting could be a promising general direction for further research in music source separation.

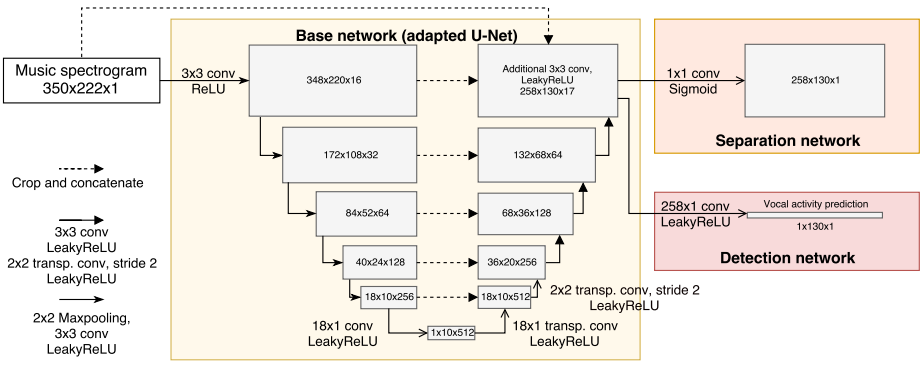


Figure 4.1: Our multi-task model for jointly detecting and separating singing voice, given the spectrogram of a music piece as input. Tensor shapes are given in the order of frequency bins, time frames, and feature channels.

4.2.1 Method

As a baseline system for SVS, we implemented a variant of the U-Net described in Section 4.2.2.2 and shown in Figure 4.1. The approach is similar to that of Stoller et al. [2018d] and Jansson et al. [2017]. A mask is predicted for a given spectrogram of a mixture excerpt. During training, audio excerpts are randomly selected from the multi-track dataset, and converted to a log-normalised spectrogram representation. The mean squared error (MSE) in spectral magnitudes between source estimates from the separator and the ground truth is used as a loss function.

4.2.1.1 Initial approach to SVS: Using additional non-vocal sections

Initially, we attempted to improve SVS performance by adding audio excerpts from instrumental sections of the SVD dataset to the SVS training set to increase its diversity. Standard supervised training on a multi-track dataset entails randomly selecting audio excerpts from the tracks to generate batches of samples. We changed this procedure so that when encountering an audio excerpt without vocals, it can be replaced with a randomly chosen non-vocal section from an additional music database with vocal activity labels. The replacement occurs with a probability of $\frac{N}{N+M}$, with N and M being the size of the SVS and SVD dataset, respectively, to ensure that non-vocal sections are effectively randomly sampled from both datasets. To train from the additional non-vocal sections, we set their target accompaniment equal to the magnitudes of the respective mixture spectrogram, and all target magnitudes of the vocal spectrogram to zero.

To evaluate separation performance, the average MSE loss shown in Equa-

tion (4.1) is measured on the test set after training the model with and without this replacement technique. We performed the above training procedure with three different set-ups for the SVS and SVD datasets.

In the first experiment, we used the DSD100 [Liutkus et al., 2017] dataset for SVS training, testing and evaluation. As SVD dataset, we combined the RWC dataset [Goto et al., 2002] (with annotations by Mauch and Dixon [2014]) and the Jamendo dataset [Ramona et al., 2008]. We also included a private collection of Dubstep, Hardstyle, Jazz, Classical and Trance music with 25 songs per genre. We found that the performance decreased compared to purely supervised learning. A first suspicion was that a bias in the test set might be responsible for inaccurate test performance measurements since only DSD100 is used (see section 4.2.1.2 for details).

To investigate this issue more closely, we conducted a second experiment and additionally included the MedleyDB [Bittner et al., 2014], CCMixter [Liutkus et al., 2015] and iKala [Chan et al., 2015] SVS datasets in the validation and test sets. Compared to the first experiment, the SVS training and test data is now less well matched, and the test performance gives a more accurate picture of generalisation capability. Here performance increased considerably using our technique, strongly indicating that a bias in the SVS training data can be alleviated by including extra non-vocal sections.

Finally, we distributed the DSD100, MedleyDB, CCMixter and iKala datasets in equal proportions into training, validation and test set for a more realistic set-up in which all available multi-track data is used, but in this experiment, separation performance again decreased using our approach.

These results suggest that the individual datasets are subject to different biases in the data distribution space, to which our approach is sensitive since it assumes that all samples come from the same distribution. These biases will be investigated in more detail in the next section. Another shortcoming of our approach is that we cannot learn from the additional vocal sections using this method since we do not have the source audio available.

4.2.1.2 Dataset bias for singing voice separation and detection

Since we are combining data from different sources, it is important to consider the impact of dataset bias on the performance of models trained on such combined data. We hypothesised that datasets used for SVD and SVS are each uniquely biased, which can include properties such as the relative energy of the sources, overall energy levels and how often vocals occur on average. We computed metrics for the above for the MedleyDB, DSD100, CCMixter, iKala, Jamendo and RWC datasets, as they are commonly used for SVD and SVS. Vocals were

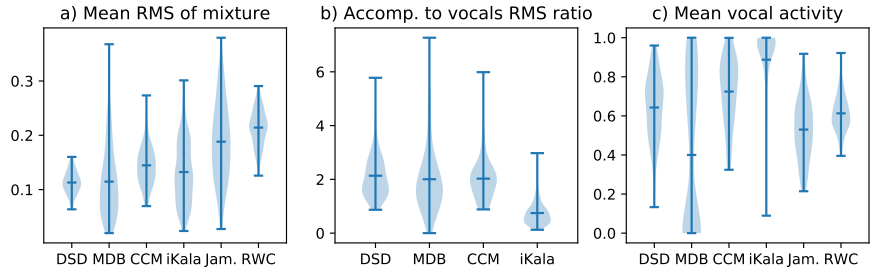


Figure 4.2: Distribution of values for different collections of tracks, for different properties. Outliers for MedleyDB in b) resulting from instrumental tracks have been excluded.

considered active if the average absolute amplitude in a 10 ms window exceeded $5 \cdot 10^{-4}$. Figure 4.2 shows the distribution of these properties for each dataset, where metrics have been averaged song-wise.

Clear dataset bias manifests itself in the uneven distribution of values across datasets. For example, iKala contains relatively loud vocals and very few instrumental sections, and CCMixter has louder tracks than DSD100 with more vocals on average. Additionally, even more dataset bias could be present in features which are more difficult to detect and quantify, such as timbre, language of the lyrics, music genre, recording conditions or the bleed level for multi-track recordings. Therefore, it is very difficult to directly prevent models from overfitting to these biases. We would like to highlight this as a critical problem for the field of SVS and SVD, since many models are trained on a single dataset source and thus may not generalise nearly as well as the test scores indicate.

4.2.1.3 Multi-task learning approach

To mitigate problems due to dataset biases, we employ a multi-task learning (MTL) approach [Caruana, 1998] instead. We augment the separation model with a component that predicts vocal activity based on a hidden layer of the separation model. We train the combined model to output the source signals in the multi-track dataset and the vocal activity labels in the SVD dataset, respectively, with most parameters being shared for both tasks.

This approach has multiple benefits. Firstly, predicting both outputs based on a shared hidden representation only assumes that the source output has some relationship with human-annotated vocal activity labels, but we do not define it explicitly. For example, temporal inaccuracy in labels could mean that the beginnings of vocals are annotated as non-vocal. If we force the vocal output of the separator to be silent for all sections annotated as non-vocal, or use the approach from Section 4.2.1.1, we give incorrect information to the separator.

Secondly, a different dataset bias for each task can be accounted for by the model to some extent with its task-specific components. Thirdly, we exploit the information present in extra non-vocal and vocal sections. Finally, the trained model can be used for both SVS and SVD.

For the SVS task, we use the MSE between the separator prediction $f_\phi(\mathbf{m})$ for a mixture excerpt $\mathbf{m}$ and the true sources $\mathbf{s}$ as the loss:

$$L_{\text{MSE}} = \mathbb{E}_{(\mathbf{m},\mathbf{s})\sim p^1} \frac{1}{N} \, ||\mathbf{s} - f_\phi(\mathbf{m})||^2 \tag{4.1}$$

where p^1 represents the multi-track dataset distribution, which is approximated by a batch of samples, and N denotes the dimensionality of the joint source vectors $\mathbf{s}$ and $f_\phi(\mathbf{m})$. For output spectrograms with T time frames, F frequency bins and K sources, $N = T \cdot F \cdot K$.

For the SVD task, we use the binary cross-entropy at each time frame of the spectrogram excerpt, averaged over time and over data points:

$$L_{\text{CE}} = \mathbb{E}_{(\mathbf{m},\mathbf{o})\sim p^2} \frac{1}{T} \sum_{t=1}^{T} \log p_\phi^t(o_t|\mathbf{m}) \tag{4.2}$$

where p_ϕ^t denotes the probability of the vocal state the model assigns to time frame t of the audio excerpt with a total of T frames, and p^2 describes the SVD dataset distribution whose samples contain a binary vector $\mathbf{o}$ with a vocal activity label o_t at each spectral frame t of the source output spectrogram.

For our MTL model, we combine the two above losses using a simple weighting scheme:

$$L_{\text{MTL}} = \alpha L_{\text{MSE}} + (1 - \alpha) L_{\text{CE}}. \tag{4.3}$$

We set $\alpha = 0.9$ so that experimentally the losses are approximately on the same scale during training. Although an optimisation of this hyper-parameter might improve results further, it is omitted here due to computational cost. We also experimented with a loss function derived from a Maximum Likelihood objective as shown in Appendix A, but did not obtain better performance.

4.2.2 Evaluation

Next, we describe the experimental evaluation procedure for our MTL approach.

4.2.2.1 Datasets

For the SVS dataset, we use DSD100 with 50 songs each for training and testing, according to the predefined split. We use the Jamendo dataset for SVD, since it predominantly contains Western Pop and Rock music, similarly to DSD100, to

avoid a large dataset bias. Jamendo's validation and test partitions comprising 30 songs are used for testing, leaving 60 songs for training. This set-up is intended as a proof of concept of the MTL approach – in this setting even slight improvements are promising, since vocal activity labels do not directly yield information on vocal structure, and should translate to larger improvements given larger SVS and particularly SVD datasets.

4.2.2.2 Model architecture and preprocessing

The audio input is converted to mono and down-sampled to 22050 Hz to reduce dimensionality, before the magnitude spectrogram is computed from a 512-point FFT with 50% overlap, and normalised by $x \rightarrow \log(1+x)$. Excerpts comprised of 222 time frames each are used as input to our model shown in Figure 4.1, which consists of a base network that branches off into a separation and a detection network.

The **base network** closely follows our previous implementation [Stoller et al., 2018d] of the U-Net [Jansson et al., 2017]. The output of an initial 3×3 convolution with 16 filters and ReLU non-linearity is fed to a down-sampling block consisting of max-pooling with size and stride two followed by a 3×3 convolution with 32 filters. The down-sampling block is applied three more times, each time doubling the number of filters, finally yielding a $18 \times 10 \times 256$ feature map. We then use a 1D convolution with filter size 18×1 before applying the respectively transposed convolution, and concatenate it with the original $18 \times 10 \times 256$ feature map to capture frequency relationships. In the following up-sampling block, a 2×2 transposed convolution with 128 filters is applied, and the output concatenated with the output of the down-sampling block at the same network depth after centre-cropping it. Lastly, a 3×3 convolution with 128 filters is applied. After applying this up-sampling block another three times, each time with half as many filters for the convolutions, the resulting $258 \times 130 \times 16$ feature map is concatenated with the centre-cropped input. The resulting features are input to the SVS as well as the SVD sub-network.

The output size is smaller than the input size since we use "valid" convolutions that do not employ implicit zero-padding. Therefore, the mixture naturally provides additional temporal context processed during convolution, and its magnitudes are zero-padded in frequency so that the separator output has the correct number of frequency bins. Unless otherwise stated, Leaky ReLU is used after all convolutions as the non-linearity to allow for better gradient flow.

In the **SVS network**, the feature map from the base architecture is transformed into a filtering mask, which is multiplied point-wise with the original mixture spectrogram magnitudes to yield the source estimates. To generate the

source audio, we use an inverse STFT using the mixture's phase, and apply 10 iterations of the Griffin-Lim algorithm [Griffin and Lim, 1984] to further refine the phase.

The **SVD network** takes the final feature map from the base architecture and applies a single $F \times 1$ filter, where F is the number of frequency bins, to reduce the time-frequency feature map to a single scalar for each time step. Application of a sigmoid non-linearity yields the probability of the presence of singing voice at each time step.

4.2.2.3 Experimental set-up and metrics

To identify the impact of our proposed approach in comparison to solving separation and detection separately, we train and evaluate our network solely for either SVS or SVD, before comparing to training with the multi-task loss.

Model performance is evaluated on the test dataset every 1000 iterations and the model with the best performance is selected. Training is stopped after 10,000 iterations without performance improvement. For SVD, we use the *area under the receiver operating characteristic (AU-ROC)* to evaluate performance. For separation, we use the MSE training objective from Equation (4.1) in the normalised magnitude space, as well as the track-wise SDR, SIR, and SAR metrics [Vincent et al., 2006] on the audio signals. We select two MTL models with the best AU-ROC and MSE values, respectively, since best performance is reached at different training stages.

4.2.2.4 Results

Table 4.1 shows a performance comparison of the considered models. For both SVD and SVS, we achieve a slight improvement in both AU-ROC and MSE performance metrics using our model variants. This is promising since the SVD dataset is small and vocal activity labels are less informative training targets than the vocals themselves. Therefore, larger datasets could be used in future work to obtain larger performance increases.

While the MSE on the normalised spectrogram magnitudes improves by about 6%, the mean SDR for vocals and accompaniment does not change significantly. To find the cause, we analyse the employed implementation for SDR computation on the DSD100 dataset[1] also used in the SiSec source separation evaluation campaign [Liutkus et al., 2017]. Tracks are partitioned into excerpts of 30s duration, using 15s of overlap, for which a local SDR value is computed. The final SDR is the average of the local SDR values. However, for excerpts where at least one source is completely silent, the SDR has an undefined value of log(0)

[1] `https://github.com/faroit/dsd100mat`

	Metric								
				Vocals			Accompaniment		
	AU-ROC	MSE	RMS	SDR	SIR	SAR	SDR	SIR	SAR
SVD	0.9239	-	-	-	-	-	-	-	-
SVS	-	0.01865	0.0194	2.83	5.27	**6.88**	6.71	**14.75**	13.25
Ours	**0.9250**	**0.01755**	**0.0155**	2.86	**5.56**	6.23	6.69	13.24	**14.11**

Table 4.1: Performance comparison between SVS and SVD baseline and our approach. "RMS" denotes the root-mean-square error of the vocal signals predicted for non-vocal sections of the mixture (lower is better). Results significantly better than the comparison model ($p < 0.05$) in bold. Significance of the AU-ROC difference determined with binary labels from all time frames as samples [DeLong et al., 1988]. A paired Wilcoxon signed-rank test was used for all other metrics.

and is excluded from the final SDR average, so that the model's performance in these sections is ignored. This is the case for 79 of 736 excerpts due to non-vocal sections and is thus a practically relevant flaw of the evaluation metric.

More sophisticated methods such a the one by Vincent [2012] take audio perception more explicitly into account, but presumably suffer from the same issue with silent sources, as similar computations are used there as well. As an ad-hoc solution, we propose computing the source estimate's energy or ideally loudness for silent sections of the source ground truth as a simple workaround and report it in addition to other metrics. Finding a consistent and perceptually accurate evaluation metric is thus an important unsolved problem, and listening tests arguably remain important to accurately assess separation quality.

A lower average MSE combined with a stagnating SDR suggests that our model improves especially on those non-vocal sections excluded from the SDR calculation. This might occur because negative vocal activity labels allow the model to learn that many different instruments present in the mixture should not be treated as vocals. As a result, the model could include less non-vocal instruments in the predicted vocal output, making it closer to a completely silent signal during non-vocal sections of the mixture. To test this hypothesis more explicitly, we take the vocal estimates of the baseline and our model and compute the average root mean square (RMS) of the 79 excerpts excluded from SDR computation, as well as the average output over whole songs in the DSD100 dataset. We find that training the model with our multi-tasking approach results in less energy in the predicted vocal output for non-vocal sections compared to training it only for SVS (see Table 4.1). This demonstrates that our model performs better on non-vocal sections and about equally on vocal sections due to a similar SDR.

4.2.3 Discussion and Conclusion

We demonstrated that jointly solving the task of singing voice detection and singing voice separation can improve performance in both tasks and alleviates the issue of dataset scarcity. Furthermore, we found biases specific to each dataset that could prevent source separation and detection models from generalising properly to unseen data. Finally, we discuss a major flaw in the most popular evaluation metric for source separation [Vincent et al., 2006] related to the performance measurement in silent sections.

Therefore, further research into improved, perceptually relevant metrics is a definite need. As a workaround, we propose additionally measuring and reporting the loudness of the model's source estimates for sections where the respective source is silent. Our multi-task approach could be generalised and applied to mixtures with pitch curve or phoneme annotations of the singing voice, or even to whole transcriptions of musical sources (see Benetos et al. [2013]). Performance increases can be expected to be larger especially for the latter case as correct predictions on one task greatly simplify solving the other one.

In the next section, we will extend our studies to more MIR tasks, to increase the generality of our findings and use meta-learning instead of multi-task learning with the goal of performing well on previously unseen MIR tasks with only little adaptation.

4.3 Meta-learning for MIR tasks

As seen in Section 4.2, solving an additional, related task can act as a prior that regularises the model to help generalisation. In this section, we extend this concept to a larger set of tasks, and also make use of meta-learning to find weight initialisations for a "universal MIR model" so that subsequent task-specific training leads to good generalisation.

Self-supervised learning [van den Oord et al., 2019] as well as transfer learning by supervised pre-training [Kong et al., 2020, Choi et al., 2018] were employed for specific audio tasks before. However, to date there is no exploration of the benefit of meta-learning applied to a variety of MIR tasks. This is particularly promising since there exist a number of well-established MIR tasks [IMIRSEL, 2020], which a meta-learning approach could directly use to obtain models that perform well on previously unseen tasks. In contrast, self-supervised learning does not make use of data labelled for different tasks at all and relies on the assumption that the self-supervised objective defined on unlabelled data imbues the model with information relevant for the actual task(s) of interest. Therefore, we will investigate the application of meta-learning to MIR in the following.

4.3.1 Method

In our approach we use generic CNN and RNN architectures as "base models" along with different "output modules" such as linear layers or attention that are task-specific and convert the base module output to the dimensionality required by the particular task. We then adapt the Reptile algorithm [Nichol et al., 2018] to find a suitable initialisation for the base module parameters, so that subsequent training on an MIR task with a certain output module quickly leads to low training error. If meta-learning is successful, it would yield a model that achieves good performance after a few training steps on a previously unseen MIR task, yielding better results than training the same model from scratch, thanks to the prior knowledge about musical structure encoded in the meta-learned weight initialisation. We will describe the details of the approach in the following.

In all of our experiments, the audio signals are first preprocessed to be at 22.05 KHz sampling rate before taking a 1024-point STFT with a hop size of 512 samples. Finally, the magnitudes of this STFT are log-normalised by applying $x \to \log(x + 1)$. This yields a $T \times F$ matrix, where T represents the number of time frames, with $F = 513$ frequency bins each, which is used as input to our model.

4.3.1.1 Model

The model used for each MIR task consists of two components: a *base module* that is shared between the different tasks and also used during pre-training, whose outputs are fed to a *task-specific module* whose parameters are randomly initialised and optimised for the particular task at hand.

As our first base module, we use a bidirectional 3-layer GRU ("BiGRU") with 512 hidden units each, as RNNs are very powerful and flexible models for tasks with sequential input. Since the model consists of a forward and backward component with 512 hidden units each, we obtain $F = 1024$ features per time-step.

As our second base module, we use a simplified U-Net architecture with 7 up- and downsampling blocks. 512 one-dimensional convolutional filters are used in each block with a temporal width of 3, producing $F = 512$ features per spectral timeframe.

The feature matrix of size $T \times F$ is then further processed by a task-specific module, whose structure depends on the task at hand. We propose three such modules that cover a wide variety of MIR tasks.

Simple output module This module simply consists of a linear layer that is applied to the feature vector at each time-frame. It is therefore suited for tasks

where temporal predictions with high temporal resolution are required, such as source separation or transcription.

GRU output module This task-specific module consists of a bidirectional GRU followed by a linear layer applied to each output, and is designed for tasks in which the predictions are typically made at a much lower resolution than that of the input spectrogram, for example when predicting the transition points between segments of a song. The module can be set to subsample the input features coming from the base module by a given factor K, before processing it using the GRU.

Global attention output module The third task-specific module makes use of attention [Bahdanau et al., 2016] to summarise the temporal features from the base module into one fixed-size feature vector of dimensionality C, which is useful for classification tasks (where C is the number of classes) such as genre, artist or mood recognition.

Specifically, two convolutions with F inputs and C outputs are applied in a convolutional fashion to the base module features over all time-steps, yielding two $T \times C$ feature matrices $\mathbf{F}^m$ and $\mathbf{F}^c$. Features in $\mathbf{F}^m$ are used to compute a mask $\mathbf{M}$ that determines how strongly the output at each timestep for each channel should be weighted in the final averaging operation over time:

$$\mathbf{M}_{i,j} = \frac{\exp \mathbf{F}^m_{i,j}}{\sum_{t=1}^{T} \exp \mathbf{F}^m_{t,j}}. \tag{4.4}$$

Afterwards, the C-dim. output vector $\mathbf{o}$ is computed as a weighted average over time-steps, where the weights are provided by the mask and the values by the content matrix $\mathbf{F}^c$:

$$\mathbf{o}_j = \sum_{t=1}^{T} \mathbf{M}_{t,j} \cdot \mathbf{F}^c_{t,j}. \tag{4.5}$$

This output can then be used as a representation of the whole input excerpt, for example as input to a cross-entropy loss for classification tasks, when C is set to the number of classes.

4.3.1.2 Meta-learning approach

Our proposed meta-learning approach is based on the Reptile algorithm [Nichol et al., 2018]. Reptile updates the weight initialisation ϕ (the meta-parameter) in each meta-training iteration (outer loop) as follows. N tasks are randomly drawn from a given set of tasks, starting with weights ϕ, for a pre-defined number of k steps (called inner loop). ϕ is then adjusted in the direction of the weights that

are obtained after task-specific training. More specifically, it is updated to

$$\phi \leftarrow \phi + \epsilon \frac{1}{N} \sum_{i=1}^{N} (\tilde{\phi}_i - \phi), \tag{4.6}$$

where $\tilde{\phi}_i$ is the weight vector obtained after training on task i. This update can effectively be implemented by using $\frac{1}{N} \sum_{i=1}^{N} (\phi - \tilde{\phi}_i)$ as the gradient in standard gradient descent but also other optimisers based on gradient descent, such as Adam.

Reptile can be straightforwardly applied to task distributions with homogenous tasks that share the same output space, since the same parametric model can be shared between all tasks. This is the case in the main application scenario presented in the original Reptile paper [Nichol et al., 2018], where all tasks involve classification into the same number of classes. However, we aim to tackle tasks with different numbers of outputs – while tagging requires summarising all temporal features into one vector of constant size equal to the number of tags, drum transcription involves making fine-grained predictions over time.

Therefore, we use task-specific output modules as outlined in Section 4.3.1.1 and we incorporate these additional, task-specific parameters into our meta-learning approach. To match the meta-training scenario with how the model is deployed at meta-test time (when training on an unseen task using the learned initialisation), we would ideally re-initialise task-specific parameters at the start of the inner loop of meta-optimisation before starting task training. However, we found that this introduces a lot of noise into the updates to ϕ due to the randomness of task-specific weight initialisation, which hinders convergence. Additionally, a large number of task training steps k is needed to adjust the task-specific parameters before informative changes can be made to the base model, which would severely slow down training. Due to the above issues, we instead re-use the task-specific parameters throughout meta-training, so that in each inner loop, task performance can reliably be improved in only a few update steps. To match actual task training conditions, we reset the task optimiser's state (such as rolling moment data) at the beginning of each inner loop.

4.3.2 Experiments

In our experiments, we pre-train models with the meta-learning objective outlined in Section 4.3.1 before using the obtained model weights as initialisation points to adapt the model to a wide variety of MIR tasks.

4.3.2.1 Tasks

We use ten MIR tasks in total to evaluate the generalisation capability of our models, which were chosen according to the following criteria.

- When possible, the dataset used for training and evaluating the model should be easily accessible by the research community.
- The task should be well established within the MIR research community as well as the corresponding evaluation metrics and procedures.
- The tasks should be relevant to the MIR community, of sufficient difficulty and cover harmonic, melodic and rhythmic aspects of music signals.

Tagging Music tagging fits our task selection criteria, as it requires the extraction of many high-level musical features such as genre, mood and instrumentation.

Our experiments generally follow the ones conducted in the "SampleCNN" work [Lee et al., 2017]. In particular, we use the Magnatagatune (MTAT) dataset with the same partitioning into training, validation and test set[2] to enable comparison with the state of the art, which focuses on predicting the 50 tags most commonly found in the whole dataset.

In contrast to the other tasks, our model processes the 30 second long audio excerpts contained in the dataset as a single input example, meaning the audio is not partitioned further before providing it to the model. To predict a 50-dimensional vector of tag probabilities for each audio excerpt, we process the base module features with the task-specific global attention output module. We use the binary cross-entropy loss independently applied to all tag probabilities, which is the standard for these multi-label prediction problems, as training loss. For validation-based early stopping and evaluation, we use the well-established AU-ROC metric.

Emotion recognition As another more abstract task similar to music tagging, we include the recognition of induced emotion. More specifically, we use the Emotify dataset [Aljanaki et al., 2016] which contains a set of songs along with ratings from multiple participants per song indicating which emotions were felt out of a total of nine emotions. We set up the task as a regression problem, using the percentage of participants indicating a particular emotion as continuous target label in the $[0, 1]$ range.

Experimentally, we broadly follow Jakubik and Kwaśnicka [2018], using the mean squared error as training objective and reporting Pearson's R on the test

[2] `https://github.com/jongpillee/music_dataset_split`

set. We use a 3-fold cross-validation, using one partition each for the training, validation and test set.

Predominant Melody Estimation Our next task is predominant melody estimation, which can be framed as predicting the fundamental frequency of the melody over time. We follow the experimental setup outlined by Bittner et al. [2017], using the same split for the MedleyDB dataset[3] and using the same "MELODY2" annotations as ground truth.

Our model uses the task-specific GRU output module to process the base model outputs and make time-dependent predictions. The 361-dimensional predictions at each time-step represent a categorical distribution over musical pitches, starting with the note C1 at 32.7 Hz, and using six octaves with 60 bins per octave resulting in 20 cents per bin and 360 pitch classes, plus one representing an "unvoiced" section. One such categorical distribution is predicted for every spectral frame in the input and therefore every 11 ms with our given hyper-parameters for the STFT computation.

For training, we use the standard cross-entropy loss used for classification tasks. To validate the model, we use the "Overall Accuracy" (OA) metric commonly used for this task [Bittner et al., 2017, 2018]. Given the model prediction, we compute this metric by taking the class with highest probability at each time-step, resampling the resulting time-dependent predictions to the sampling rate of the annotation, converting the pitch classes to cent values, and processing the result with the respective evaluation functions[4] from the "mir_eval" toolbox [Raffel et al., 2014].

Drum transcription To cover the rhythmic dimension of music, we include the drum transcription task into our task suite. Following the experimental setup outlined in Vogl et al. [2017], the task is to locate the onset times of hi-hat, snare and kick drum hits in music recordings. In our experiments, we focus on the "ENST-Drums" dataset [Gillet and Richard, 2006] as a well established dataset in the drum transcription literature.

Our model uses the task-specific GRU output module to convert the base module features into three predicted probabilities at each time-step, one for each drum instrument. For training, we apply the binary cross-entropy loss independently to every time-frame and output class, representing a multi-label task at each time-step. For validation-based early stopping and evaluation, we

[3] `https://github.com/rabitt/ismir2017-deepsalience/blob/b25e758c04f2313159094c7d29eeabef0665b5bf/outputs/data_splits.json`

[4] We use "resample_melody_series" for resampling, "to_cent_voicing" for conversion to cents, and "overall_accuracy" to compute the overall accuracy (OA).

use the standard F-Measure with a tolerance window of 20ms, using an activation threshold that maximises the resulting F-Measure.

Segmentation We tackle the problem of music segmentation, or more specifically, detecting the boundaries between musical segments as done by Grill and Schlüter [2015]. The difficulty of this task arises due to its high-level nature (requiring features with a high degree of abstraction such as time-dependent indicators of mood, instrumentation or motifs), and since it is concerned with long-term dependencies in the music piece such as the repetitions of segments.

The "SALAMI" dataset [Smith et al., 2011] is used for our experiments, which contains a large collection of mostly live music recordings along with timestamps of segment boundaries. We remove start and end boundaries that lie very close to the beginning and end of the music piece.

For our model, we use the task-specific GRU output module to transform the base module features into a one-dimensional vector indicating the probability of a segment boundary at each time-step of the input spectrogram. The output module is set to a downsampling factor of 43, meaning one prediction is made for every 23 input steps, resulting in one prediction every $\frac{43 \cdot 512}{22050} \approx 1$ second. As training loss, we use the binary cross-entropy loss as an average over all time-steps. For validation-based early stopping and evaluation, we use the F-Measure with a tolerance window of 3 seconds.

Music source separation Our next task we consider is music source separation. Since it is usually framed as a regression problem, the output dimensionality of the model is very large, and the output needs to be on a very high temporal resolution. This is in contrast to previous work on self-supervised, meta-learning and multi-task learning, which has been focused largely on classification problems, and so presents a particular challenge in our context.

For our dataset, we use the MUSDB18 dataset since it is used in the SiSec music source separation challenge and therefore offers direct comparison to current state-of-the-art methods. As task-specific module, we use the GRU output module to process the base module features. To train our model, we use the L1 norm of the difference between the predicted and the true spectrogram magnitudes, and for validation as well as evaluation we use the standard SDR, SIR and SAR metrics [Vincent et al., 2006] expressed in dB.

Tempo estimation As another rhythmic task, we include tempo estimation on the "Giantsteps Tempo" dataset [Knees et al., 2015]. While similar to beat detection, deriving an accurate global tempo estimate can require aggregating rhythmic information over the whole music input in the presence of tempo

fluctuations. We use a 3-fold cross validation, using one partition each for the training, validation and test set.

We discretise the continuous tempo labels in beats per minute by assigning them to the nearest of 50 tempo categories with tempo $50 \cdot 1.03^i$ for category $i \in \{0, \ldots, 49\}$. Following the approach by Schreiber [2018], this allows us to cast the problem as classification using a cross-entropy loss. For evaluation, we use accuracy based on the above defined tempo classes ("Accuracy0"), as well as "Accuracy1" which counts an estimated tempo as correct if it deviates at most 4% from the ground truth tempo in either direction.

Singing voice detection We include singing voice detection since it is a very established task in the MIR community [Lee et al., 2018], using the popular "Jamendo" dataset [Ramona et al., 2008].

We convert the onset and offset times of singing voice annotated in the dataset to binary sequences over time according to our model's temporal output rate, with a target label, indicating presence of singing voice with a 1 and absence with a 0. To train the model, we use the binary cross-entropy averaged over all output time-frames. For evaluation, we use the area under the ROC-curve (AUROC measure) to estimate performance independent of the chosen classification threshold.

Beat detection As a related task to drum transcription and tempo estimation, we use beat and downbeat detection as another rhythm-oriented task in our task suite. We use the "GTZAN" dataset [Tzanetakis and Cook, 2002] for our experiments, using the beat annotations from Marchand et al. [2015]. We use a 3-fold cross-validation, using one partition each for the training, validation and test sets.

The simple output model with two output channels is used as a task-specific module to estimate probabilities of down-beat and beat occurring at each input time-step. Binary cross-entropy is used as a training objective, averaged over both output channels and input time-steps. To produce predictions, we use a simple peak picking scheme, whose parameters are optimised on the validation set (see Section 4.3.2.2 for details). For evaluation, we use the (micro) F-Measure, with a tolerance window of 0.07 seconds, following the approach by Böck et al. [2016].

Key detection Detecting the musical key is another important basic task within MIR, so we include it as a 24-way classification problem (12 halftones as tonic each with a major and minor mode) in our experiments. We use the "GiantstepsKey" dataset [Knees et al., 2015] to train our model, using the global

attention module as task-specific output module. We use a 3-fold cross-validation, using one partition each for the training, validation and test set. For evaluation we simply use accuracy as evaluation metric.

4.3.2.2 Training setup

Meta-learning We partition our set of tasks into two sets of five, ensuring that similar tasks are in different partitions. Set 1 contains beat detection, source separation, SVD, tagging and tempo detection, while set 2 is comprised of drum transcription, emotion and key detection, melody estimation and segmentation. We use this partitioning to evaluate our meta-learning approach in a cross-validation setting: one set is used as the set of training tasks to learn an optimal weight initialisation from, before using it for training on the tasks from the other set to test the performance of our approach on unseen tasks.

To optimise the meta-training objective, we use an ADAM optimiser with learning rate 10^{-3} and a total of 10000 training iterations. We linearly anneal the meta learning rate towards zero over the course of training. We sample $N = 3$ tasks in each outer loop iteration to obtain a more stable meta-gradient, especially as we feature a very diverse set of tasks.

Task training We perform task-specific training with and without pre-training the base module. Early stopping is employed using a loss computed on an additional validation set to prevent over- or under-training the model, enabling us to ascertain whether pre-training is effective – if it is, we would for example expect the early stopping to occur sooner with than without pre-training.

Peak picking For segmentation, drum transcription, and beat detection, we train our model predict onset probabilities at each time-step, but these have to be converted into a sequence of detected events to obtain predictions. For this, peak picking is performed on the probability sequences obtained from the model. Specifically, local maxima are identified as peaks if they exceed a certain probability threshold t, have a certain minimum distance d to other peaks, and have a minimum "prominence" p – see the "find_peaks" method from the "scipy" library[5] for more details on these parameters.

The above three peak-picking parameters are optimised on the validation set by a simple grid search to maximise micro F-Measure.

[5] `https://scipy.org/scipylib/`

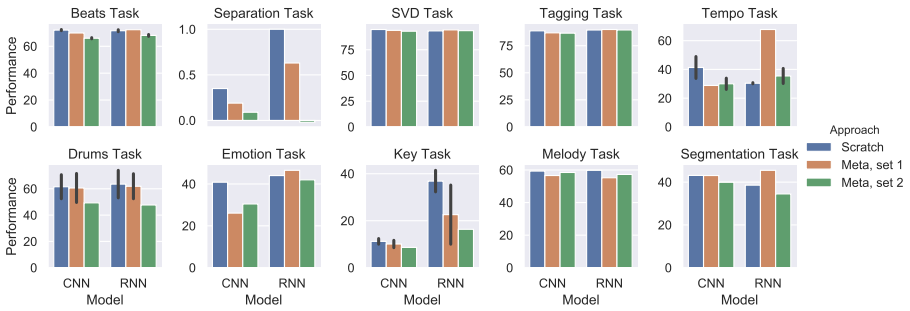


Figure 4.3: Results of our meta-learning experiments for different network architectures and tasks, comparing meta-learned weight initialisations to random ones (trained "from scratch"). The top and bottom rows correspond to task sets 1 and 2, respectively. Performance metrics for each task are described in Section 4.3.2.1 (higher is better). For configurations with cross-validation, bars indicate the mean and and error bars the standard deviation computed across the folds. Note that some configurations do not feature cross-validation to avoid using test data for training.

4.3.3 Results

We evaluate the performance of the CNN and RNN model on the ten tasks outlined in Section 4.3.2.1, when training the models from a random weight initialisation ("from scratch"), or from an initialisation found by meta-learning on the task set 1 or 2.

The results for all configurations are shown in Figure 4.3. Overall, meta-learning appears to provide mixed results in our experimental setting. Oftentimes, performance is unchanged or even lower when using meta-learning compared to simply training the model from scratch. We suspect that this might be due to the number of tasks used for meta-training, which is unusually low compared to other applications such as few-shot learning [Finn et al., 2017], where many tasks are automatically constructed. Additionally, our tasks are arguably also more dissimilar, since they feature different datasets and output spaces. The small dataset of tasks in combination with the task diversity makes meta-learning very prone to overfitting to the set of training tasks.

However, there are some cases in which meta-learning improves performance, most notably on the tempo detection and segmentation tasks. On these tasks, meta-learning on the task set 1 provides substantial increases in performance when using the RNN model, and even outperforms all other tested configurations[6]. For the tempo task in particular, the beat detection task included in set 1 might inform the model about the correct metrical level for a song to avoid mistakenly

[6]These improvements are both significant as shown by paired Wilcoxon tests comparing sample-wise F-Measures of both models for segmentation ($p \approx 4.8 \cdot 10^{-16}$) and sample-wise binary success outcomes of both models (on the first fold) for tempo estimation ($p \approx 1.1 \cdot 10^{-16}$)

doubling or halving the tempo prediction.

Overall, these results demonstrate that meta-learning can, in some scenarios, imbue models with prior knowledge about musical structure to enhance performance. However, this effect is not observed in the same context when using the CNN model, suggesting that model structure has an important effect on meta-learning outcomes. Due to the large diversity of tasks we consider, the CNN model might not be sufficiently flexible to transfer knowledge between tasks with significantly different output structure. On the emotion detection task, meta-learning appears to help performance as well when using set 1, however when using the paired Wilcoxon test to compare the series of sample-wise MSE values achieved by both models on the test set, we did not find a significant difference between the two models ($p = 0.82$).

Furthermore, the datasets used for some of the tasks might be "saturated", in the sense that label noise or the simplicity of the task means that additional prior knowledge does not help the model with prediction. For example, we found label noise to be present in the tagging task, since oftentimes songs do not get assigned to all the matching tags, but only a subset of them. Still, we achieve an AU-ROC of 89.51 by training the RNN from scratch, an impressive performance on par with current state of the art approaches, e.g. Lee et al. [2017] reaching a maximum performance of 90.55 in the same experimental conditions. Singing voice detection on the other hand is a relatively simple task, so that not much annotated data appears to be needed to reach good performance.

4.3.4 Discussion and Conclusion

In the above section, we investigated meta-learning as a potential approach to improve generalisation of models for MIR tasks, since this has not been explored in previous literature, and many MIR tasks do not feature large datasets. Our results indicate that meta-learning has some potential in the field of MIR, with substantial performance increases in some cases compared to training models from scratch on the task's dataset. However, in many cases performance actually decreases (which is called "negative transfer"). We suspect this happens due to one or more of the following reasons:

- Overfitting during meta-learning due to the small number and high diversity of training tasks;
- Choice of DNN architecture – too many or too few parameters, or not flexible enough to adapt to different task types (classification, time-varying outputs);

- Inner-loop optimisation procedure during meta-learning not equivalent to how the model is trained at test time.

Therefore, future work is required to obtain more consistent performance improvements using meta-learning. One fundamental step in this direction would be the development of a large collection of MIR tasks where tasks and annotations are offered in a standardised format and so can be easily used by researchers. To simplify the evaluation of meta-learning approaches, it would be useful to have a single evaluation metric that aggregates all per-task performance metrics into one number. A major challenge here is that performance metrics differ between MIR tasks – while some classification tasks such as tagging and tempo estimation could be evaluated under the same criteria (e.g. classification accuracy), source separation for example uses SDR values that behave very differently.

Another difficulty in our experiments is the different complexities of tasks and the varying amounts of training data available for each task. Ideally, this should be accounted for by the meta-learning approach, for example by using an adaptive number of training iterations per task that grows with task complexity and dataset size. In general, we used only 5 training steps in the inner loop of meta-learning to simulate the results of training on a particular task at test time – a value inspired by the most common application of meta-learning found in literature so far: few-shot learning [Finn et al., 2017]. However, at test time hundreds or thousands of steps were often needed in our experiments to achieve optimal performance. This mismatch between train and test time is almost certainly detrimental to the performance of our meta-learning approach, as it optimises for the performance achieved after only 5 training steps. Since the overall time needed for meta-learning is the product of meta-training steps, inner loop steps, and the number of tasks sampled at each meta-training step, it is very computationally costly to increase the number of inner loop iterations. Therefore, future work is needed to investigate other approaches with better time complexity.

4.4 Conclusion

In this chapter, we developed data-driven priors for MIR models, with the goal of imbuing these models with musical knowledge to improve their generalisation capabilities, especially when the amount of labelled training data is limited. To achieve this, our priors integrate information from multiple related MIR tasks, so that performance on some set of target tasks is improved, following methods from multi-task learning [Caruana, 1998] and meta-learning [Finn et al., 2017].

Note that other approaches, such as self-supervised learning [Chen et al., 2020, Pascual et al., 2019, Trinh et al., 2019], also present a promising avenue for future work to build powerful audio and music priors.

In our initial study in Section 4.2, we explore the potential of multi-task learning for MIR, since it features a large number of inter-related tasks. We choose singing voice separation and singing voice detection as two particularly closely related tasks, and train a U-Net model to perform both tasks simultaneously. We observe a performance increase for both tasks when the model is trained with the proposed multi-task learning approach, which suggests that including more MIR tasks might improve performance further, given that the tasks share enough similarity. However, MTL does not provide guarantees on how trained models will perform on MIR tasks not seen during training, as it does not learn how to quickly adapt the model at test time to a given new task. Therefore, our second, larger study in Section 4.3 features ten MIR tasks with the goal of encoding more generally applicable music knowledge into the model, and uses meta-learning to find a suitable weight initialisation so that subsequent training on a particular task yields good performance after few training iterations. In contrast to multi-task learning, meta-learning can produce models that also perform well on unseen tasks, given that the distributions of training tasks and test tasks are sufficiently similar. In this study, we observe that the meta-learning approach frequently does not significantly improve the results compared to training the same model from scratch. However, there are a few notable exceptions, where performance improvements are surprisingly large, for example in the case of tempo detection. While the results are not particularly impressive, this study represents the first attempt at applying meta-learning approaches to MIR models, which will hopefully spur more research in this area and also provide initial baseline results that such future work can use for comparison purposes.

In this chapter, we investigated how to incorporate data-driven priors into models. To apply such data-driven priors on larger scales (e.g. involving longer pre-training procedures), we will design efficient, end-to-end models for high-dimensional sequence inputs such as time-domain audio signals in the following chapter.

Chapter 5

Efficient end-to-end models for temporal data

5.1 Motivation

Feeding training examples directly to "end-to-end" deep models can be beneficial for model performance since all model components can be jointly optimised towards the target objective. This is demonstrated by the success of DNNs in computer vision [Krizhevsky et al., 2012] that process images directly instead of operating on image features. However, without feature preprocessing, input dimensionality is often very large for high-resolution data encountered in many application domains – for example, raw audio data is commonly sampled at 44100 samples per second. With a large input dimensionality and parameter count, end-to-end DNN models in these domains are very computationally expensive and also tend to exhibit more overfitting due to the curse of dimensionality.

Due to the above problems, data points are very commonly pre-processed by extracting relevant features. This reduces the input dimensionality of the model and improves its generalisation since it can no longer mistakenly pick up on irrelevant features (noise) in the training data. However, determining which features are relevant (feature design) is very difficult and requires prior knowledge that is not always available. Furthermore, the performance of such models appears to fall behind end-to-end approaches as more labelled data becomes available[1].

In the presence of limited labelled data, techniques to incorporate additional related data can be applied to large models with little or no feature preprocessing to improve performance, as demonstrated in Sections 3 and 4. In contrast to

[1] `http://www.incompleteideas.net/IncIdeas/BitterLesson.html`

hand-crafted feature engineering for a small model, this approach is more scalable as these techniques act as a data-driven model prior whose power increases with the availability of more data.

To make this approach more usable, we will present novel end-to-end DNN models that are more computationally efficient by using a carefully designed architecture. For machine listening applications specifically, we present a model that does not rely on the feature preprocessing usually used in the field. More specifically, input audio is almost always first transformed into some kind of time-frequency representation, normally by taking the magnitudes from a Short-Time Fourier Transform (STFT) before possibly applying more feature engineering to reduce its dimensionality. While this can help in developing models with less prohibitive computational requirements, it also leads to two problems. Firstly, the phase information is discarded, which could be relevant for solving the task at hand, thus limiting performance. This is especially the case for tasks like audio source separation, where even small details in the audio input should be reflected in the predicted audio output. Secondly, parameters of the spectral transformation are almost always fixed, so they are likely sub-optimally chosen for the task. Defining them as hyper-parameters avoids this problem, but incurs a heavy computational overhead, since hyper-parameter optimisation is slow in this setting due to the long training time of models. Therefore, it would be desirable to optimise all parameters jointly instead.

When using raw audio input to avoid the above issues with spectrogram representations, we face the challenge of capturing long-term dependencies. In general, capturing long-term dependencies in tasks such as audio source separation, but also in other domains such as audio or text generation is important: What occurs in the audio signal at one point in time can still affect its content seconds or minutes later. Similarly, the content of a paragraph in a novel can often affect the choice of words much later down the line, which is important to model for text generation. But for sequence data sampled at high temporal resolutions such as raw audio, these long-term dependencies can span hundreds of thousands of timesteps, so it is difficult to model them efficiently.

In this chapter, we will present deep learning models for such high-dimensional sequential data that are end-to-end, i.e. directly map the raw input to the target output without relying on feature engineering, and also computationally efficient despite the high input dimensionality. The end-to-end architecture employs 1D convolutions across the sequential input and gains computational efficiency by incorporating the slow feature hypothesis [Wiskott and Sejnowski, 2002], which states that for a wide variety of tasks, important features of an input signal vary only slowly over time. As a result, the computations in many layers can be performed very sparsely across time (infrequently).

We will present a non-causal model that implements the above ideas ("Wave-U-Net") in Section 5.2 which we apply experimentally to an audio source separation problem. To adapt the model to auto-regressive sequence modelling, where the next element in a sequence needs to be predicted given *only* the previous ones, we change the convolutions to be causal to prevent the model from accessing future elements. The resulting "Seq-U-Net" is presented in Section 5.3, where we show its efficacy in a range of sequence modelling tasks including symbolic music as well as text and raw audio generation.

5.2 Wave-U-Net

Current methods for audio source separation almost exclusively operate on spectrogram representations of the audio signals [Huang et al., 2014, Jansson et al., 2017], as they allow for direct access to components in time and frequency. In particular, after applying a short-time Fourier transform (STFT) to the input mixture signal, the complex-valued spectrogram is split into its magnitude and phase components. Then only the magnitudes are input to a parametric model, which returns estimated spectrogram magnitudes for the individual sound sources. To generate corresponding audio signals, these magnitudes are combined with the mixture phase and then converted with an inverse STFT to the time domain. Optionally, the phase can be recovered for each source individually using the Griffin-Lim algorithm [Griffin and Lim, 1984].

This approach has several limitations. Firstly, the STFT output depends on many parameters, such as the size and overlap of audio frames, which can affect the time and frequency resolution. Ideally, these parameters should be optimised in conjunction with the parameters of the separation model to maximise performance for a particular separation task. In practice, however, the transform parameters are fixed to specific values. Secondly, since the separation model does not estimate the source phase, it is often assumed to be equal to the mixture phase, which is incorrect for overlapping partials. Alternatively, the Griffin-Lim algorithm can be applied to find a signal whose magnitudes are equal to the estimated ones, but this is slow and often no such signal exists [Le Roux et al., 2008], so the result is only an approximation. Lastly, the mixture phase is ignored in the estimation of sources, which can potentially limit the performance. Thus, it would be desirable for the separation model to learn to estimate the source signals including their phase directly.

In an attempt to tackle the above problems, several audio processing models were recently proposed that operate directly on time-domain audio signals [Luo and Mesgarani, 2017, van den Oord et al., 2016]. This includes approaches for speech denoising, which is a task that is closely related to the general audio

source separation problem [Pascual et al., 2017, Rethage et al., 2017]. Inspired by these first results, we investigate the potential of fully end-to-end time-domain separation systems in the face of unresolved challenges. In particular, it is not clear if such a system will be able to deal effectively with the very long-range temporal dependencies present in audio due to its high sampling rate. Further, it is not obvious upfront whether the additional phase information will indeed be beneficial for the task, or whether the noisy phase might be detrimental for the learning dynamics in such a system. Overall, we make the following contributions.

- We propose the Wave-U-Net[2], a one-dimensional adaptation of the U-Net architecture [Ronneberger et al., 2015, Jansson et al., 2017], which separates sources directly in the time domain and can take large temporal contexts into account.
- We show a way to provide the model with additional input context to avoid artifacts at the boundaries of output windows, in contrast to previous work [Pascual et al., 2017, Jansson et al., 2017].
- We replace strided transposed convolution used in previous work [Jansson et al., 2017, Pascual et al., 2017] for upsampling feature maps with linear interpolation followed by a normal convolution to avoid artifacts.
- The Wave-U-Net achieves good multi-instrument and singing voice separation, the latter of which compares favourably to our re-implementation of the state-of-the-art network architecture [Jansson et al., 2017], which we train under comparable settings.
- Since the Wave-U-Net can process multi-channel audio, we compare stereo with mono source separation performance.
- We highlight an issue with the commonly used Signal-to-Distortion ratio evaluation metric, and propose a work-around.

It should be noted that we expect the current state of the art model as presented by Jansson et al. [2017] to yield higher separation quality than what we report here, as the training dataset used by Jansson et al. [2017] is well-designed, highly unbiased and considerably larger. However, we believe that our comparison with a re-implementation trained under similar conditions might be indicative of relative performance improvements, as both models should scale similarly with increasing training set size.

[2]Code available at `https://github.com/f90/Wave-U-Net`

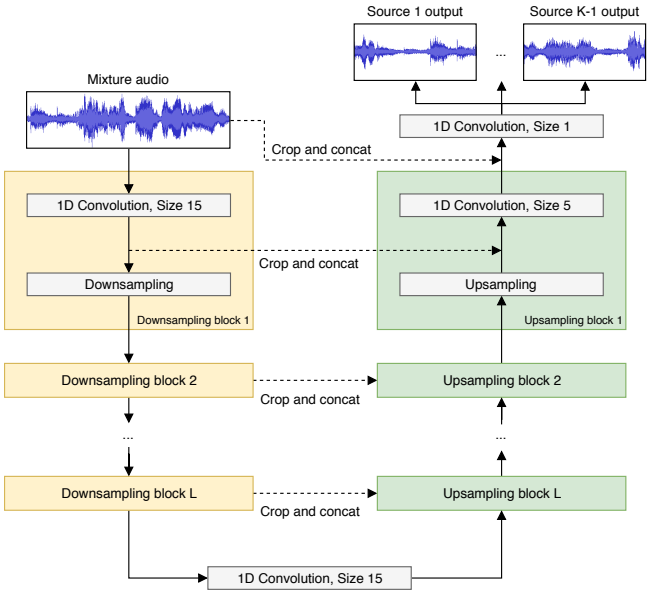


Figure 5.1: Our proposed Wave-U-Net with K sources and L layers. With our difference output layer, the K-th source prediction is the difference between the mixture and the sum of the other sources (not shown).

5.2.1 Model

Our goal is to separate a mixture waveform $\mathbf{M} \in [-1,1]^{L_m \times C}$ into K source waveforms $\mathbf{S}^1, \ldots, \mathbf{S}^K$ with $\mathbf{S}^k \in [-1,1]^{L_s \times C}$ for all $k \in \{1, \ldots, K\}$, C as the number of audio channels and L_m and L_s as the respective numbers of audio samples. For model variants with extra input context, we have $L_m > L_s$ and make predictions for the centre part of the input.

5.2.1.1 The base architecture

A diagram of the Wave-U-Net architecture is shown in Figure 5.1. It computes an increasing number of higher-level features on coarser time scales using downsampling (DS) blocks. These features are combined with the earlier computed local, high-resolution features using upsampling (US) blocks, yielding multi-scale features which are used for making predictions. The network has L levels in total, with each successive level operating at half the time resolution of the previous one. For K sources to be estimated, the model returns predictions in the interval $(-1, 1)$, one for each source audio sample.

The detailed architecture is shown in Table 5.1. `Conv1D`(x,y) denotes a 1D convolution with x filters of size y. It includes zero-padding for the base architecture, and is followed by a LeakyReLU activation (except for the final one, which uses tanh). `Decimate` discards features for every second time step to

Block	Operation	Shape
	Input	$(16384, 1)$
DS, repeated for $i = 1, \ldots, L$	`Conv1D`$(F_c \cdot i,\ f_d)$	
	`Decimate`	$(4, 288)$
	`Conv1D`$(F_c \cdot (L+1),\ f_d)$	$(4, 312)$
US, repeated for $i = L, \ldots, 1$	`Upsample`	
	`Concat`(DS block i)	
	`Conv1D`$(F_c \cdot i,\ f_u)$	$(16834, 24)$
	`Concat`(Input)	$(16834, 25)$
	`Conv1D`$(K,\ 1)$	$(16834, 2)$

Table 5.1: Block diagram of the base architecture. Shape describes the final output after potential repeated application of blocks and denote the number of time steps and feature channels, in that order. Downsampling (DS) block i refers to the output before decimation. Note that the Upsampling (US) blocks are applied in reverse order, from level L to 1. F_c is a scalar hyper-parameter that scales the number of convolutional filters in each layer. f_d and f_u are hyper-parameters determining the size of the convolutional filters in the DS and US blocks, respectively.

halve the time resolution. `Upsample` performs upsampling in the time direction by a factor of two, for which we use linear interpolation (see Section 5.2.1.2 for details). `Concat`(x) concatenates the current, high-level features with more local features x. In extensions of the base architecture (see below), where `Conv1D` does not involve zero-padding, x is centre-cropped first so it has the same number of time steps as the current layer.

5.2.1.2 Avoiding aliasing artifacts due to upsampling

Many related approaches use transposed convolutions with strides to upsample feature maps [Pascual et al., 2017, Jansson et al., 2017]. This can introduce aliasing effects in the output, as shown for the case of image generation networks [Odena et al., 2016]. In initial tests, we also found artifacts when using such convolutions as upsampling blocks in our Wave-U-Net model in the form of high-frequency buzzing noise.

Transposed convolutions with a filter size of k and a stride of $x > 1$ can be viewed as convolutions applied to feature maps padded with $x - 1$ zeros between each original value [Dumoulin and Visin, 2016]. We suspect that the interleaving with zeros without subsequent low-pass filtering introduces high-frequency patterns into the feature maps, shown symbolically in Figure 5.2, which leads to periodic noise in the final output as well. Instead of transposed strided convolutions, we perform an upsampling step, which ensures temporal continuity in the feature space, followed by a normal convolution. Although upsampling can be implemented by interleaving the signal with zeros before

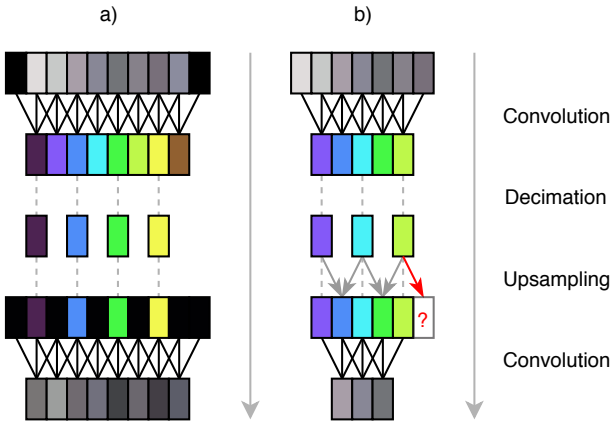


Figure 5.2: a) Common model (e.g. [Jansson et al., 2017]) with an even number of inputs (grey) which are zero-padded (black) before convolving, creating artifacts at the borders (dark colours). After decimation, a transposed convolution with stride 2 is shown here as upsampling by zero-padding intermediate and border values followed by normal convolution, which likely creates high-frequency artifacts in the output. b) Our model with proper input context and linear interpolation for upsampling from Section 5.2.1.5 does not use zero-padding. The number of features is kept uneven, so that upsampling does not require extrapolating values (red arrow). Although the output is smaller, artifacts are avoided.

low-pass filtering, ideally by convolving with a sinc filter, this is expensive to compute, so we use linear interpolation instead as an approximation. In initial tests, we achieved similar performance as measured by the signal-to-distortion (SDR) metric compared to using transposed convolutions, while avoiding sound artifacts we observed from the use of transposed convolutions.

5.2.1.3 Architectural improvements

Table 5.1 describes the architecture of the baseline variant of the Wave-U-Net (Model M1), which uses zero-padding for convolutions and K output convolutions for K sources. In the following, we will describe a set of architectural improvements designed to increase model performance.

5.2.1.4 Difference output layer

Our baseline model M1 outputs one source estimate for each of K sources by independently applying K convolutional filters followed by a tanh non-linearity to the last feature map. In the separation tasks we consider, the mixture signal is the sum of its source signal components: $\mathbf{M} = \sum_{j=1}^{K} \mathbf{S}^j$. Since our baseline model is not constrained in this fashion, it has to learn this rule approximately

to avoid highly improbable outputs, which could slow down learning and reduce performance. Therefore, we use a difference output layer to constrain the outputs $\hat{\mathbf{S}}^j$, enforcing $\sum_{j=1}^{K} \hat{\mathbf{S}}^j = \mathbf{M}$: only $K-1$ convolutional filters with a size of 1 are applied to the last feature map of the network, followed by a tanh non-linearity, to estimate the first $K-1$ source signals. The last source is then simply computed as $\hat{\mathbf{S}}^K = \mathbf{M} - \sum_{j=1}^{K-1} \hat{\mathbf{S}}^j$.

This type of output was also used for speech denoising [Rethage et al., 2017] as part of an "energy-conserving" loss, and a similar idea can be found very commonly in spectrogram-based source separation in the form of masks that distribute the energy of the input mixture magnitudes to the output sources. Introducing this layer to the baseline model (leading to model M2) should improve performance in our case, as its additivity assumption is satisfied by our dataset.

5.2.1.5 Prediction with proper input context and resampling

In previous work [Jansson et al., 2017, Grais et al., 2018, Pascual et al., 2017], the input and the feature maps are padded with zeros before convolving, so that the resulting feature map does not change in its dimension, as shown in Figure 5.2a. This simplifies the network's implementation, since the input and output dimensions are the same. Zero-padding audio or spectrogram input this way effectively extends the input using silence at the beginning and end. However, taken from a random position in a full audio signal, the information at the boundary becomes artificial, i.e. the temporal context for this excerpt is given in the full audio signal but is ignored and assumed to be silent. Without proper context information, the network thus has difficulty predicting output values near the beginning and end of the sequence. As a result, simply concatenating the outputs as non-overlapping segments at test time to obtain the prediction for a full audio signal can create audible artifacts at the segment borders, as neighbouring outputs can be inconsistent when they are generated without correct context information. In Section 5.2.3.2, we investigate this behaviour in practice.

As a solution, we employ convolutions without implicit padding and instead provide a mixture input larger than the size of the output prediction, so that the convolutions are computed on the correct audio context (see Figure 5.2b). This change is applied to model M2, yielding model variant M3. Due to the reduced feature map sizes, we constrain the possible output sizes of the network so that feature maps are always large enough for the following convolution.

Further, when resampling feature maps, feature dimensions are often exactly halved or doubled [Jansson et al., 2017, Pascual et al., 2017], as shown in

Figure 5.2a for transposed strided convolution. However, this necessarily involves extrapolating at least one value at a border, which can again introduce artifacts. Instead, we interpolate only between known neighbouring values and keep the very first and last entries, producing $2n - 1$ entries from n or vice versa, as shown in Figure 5.2b. To recover the intermediate values after decimation, while keeping border values the same, we ensure that feature maps have odd dimensionality.

5.2.1.6 Stereo channels

To accommodate for multi-channel input with C channels, we simply change the input $\mathbf{M}$ from an $L_m \times 1$ to an $L_m \times C$ matrix. Since the second dimension is treated as a feature channel, the first convolution of the network takes into account all input channels. For multi-channel output with C channels, we modify the output component to have K independent convolutional layers with filter size 1 and C filters each. With a difference output layer, we only use $K - 1$ such convolutional layers. We use this simple approach with $C = 2$ to perform experiments with stereo recordings and investigate the degree of improvement in source separation metrics when using stereo instead of mono estimation. It is added to model variant M3 and denoted as model M4.

5.2.1.7 Learned upsampling for Wave-U-Net

Linear interpolation for upsampling is simple, parameterless and encourages feature continuity. However, it may be restricting the network capacity too much. Perhaps, the feature spaces used in these feature maps are not structured so that a linear interpolation between two points in feature space is a useful point on its own, so that a learned upsampling could further enhance performance. To this end, we propose the learned upsampling layer. For a given $F \times n$ feature map with n time steps, we compute an interpolated feature $f_{t+0.5} \in \mathbb{R}^F$ for pairs of neighbouring features $f_t, f_{t+1} \in \mathbb{R}^F$ using parameters $w \in \mathbb{R}^F$ and the sigmoid function σ to constrain each $w_i \in w$ to the $[0, 1]$ interval:

$$f_{t+0.5} = \sigma(w) \odot f_t + (1 - \sigma(w)) \odot f_{t+1} \tag{5.1}$$

This can be implemented as a 1D convolution across time with F filters of size two and no padding with a properly constrained matrix. The learned interpolation layer can be viewed as a generalisation of simple linear interpolation, since it allows convex combinations of features with weights other than 0.5. It is added to model M4, obtaining model variant M5.

5.2.2 Experiments

We evaluate the performance of our models on two tasks: Singing voice separation and music separation with bass, drums, guitar, vocals and "other" instruments as categories, as defined by the SiSec separation campaign [Liutkus et al., 2017].

5.2.2.1 Datasets

75 tracks from the training partition of the MUSDB [Rafii et al., 2017] multi-track database are randomly assigned to our training set, and the remaining 25 tracks form the validation set, which is used for early stopping. Final performance is evaluated on the MUSDB test partition comprised of 50 songs. For singing voice separation, we also add the whole CCMixter database [Liutkus et al., 2015] to the training set.

As data augmentation for both tasks, we multiply source signals with a factor chosen uniformly from the interval $[0.7, 1.0]$ and set the input mixture as the sum of source signals. No further data preprocessing is performed, only a conversion to mono (except for stereo models) and downsampling to 22050 Hz.

5.2.2.2 Training procedure

During training, audio excerpts are sampled randomly and inputs padded accordingly for models with input context. As loss, we use the mean squared error (MSE) over all source output samples in a batch. We use the ADAM optimizer with learning rate 0.0001, decay rates $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and a batch size of 16. We define 2000 iterations as one epoch, and perform early stopping after 20 epochs of no improvement on the validation set, measured by the MSE loss. Afterwards, the last model is fine-tuned further, with the batch size doubled and the learning rate lowered to 0.00001, again until 20 epochs without improvement in validation loss. Finally, the model with the best validation loss is selected.

5.2.2.3 Model settings

For our baseline model M1, we use $L_m = L_s = 16384$ input and output samples, $L = 12$ layers, $F_c = 24$ extra filters per layer and filter sizes $f_d = 15$ and $f_u = 5$. Since model M4 performs best ouf of all models M1 to M5, we also apply it to multi-instrument separation (M6). Models with input context (M3 to M6) have $L_m = 147443$ input and $L_s = 16389$ output samples.

For comparison with previous work, we also train the spectrogram-based U-Net architecture [Jansson et al., 2017] (U7) that achieved state-of-the-art vocal separation performance, and a Wave-U-Net comparison model (M7) under the same conditions, both using the audio-based MSE loss and mono signals

downsampled to 8192 Hz. M7 is based on the best model M4, but is set to $L_m = 233459$ and $L_s = 102405$ to have very similar output size compared to U7 ($L_s = 98650$ samples), $F_c = 34$ to bring our network to the same size as U7 (20M param.), and the initial batch size is set to four due to the high amount of memory needed per sample. To train U7, we backpropagate the error through the inverse STFT operation that is used to construct the source audio signal from the estimated spectrogram magnitudes and the mixture phase. We also train the same model with an L1 loss on the spectral magnitudes (U7a), following [Jansson et al., 2017]. Since the training procedure and loss are exactly the same for networks U7 and M7, we can fairly compare both architectures by ensuring that performance differences do not arise simply because of the amount of training data or the type of loss function used, and also compare with a spectrogram-based loss (U7a). Despite our effort to enable an overall model comparison, note that some training settings such as learning rates used by [Jansson et al., 2017] might differ from ours (and are partly unknown) and could provide better performance with U7 and U7a than shown here, even with the same dataset.

5.2.3 Results

In the following, we will present quantitative results that enable objective model comparison, before discussing some qualitative observations.

5.2.3.1 Quantitative results

The signal-to-distortion (SDR) metric is commonly used to evaluate source separation performance [Vincent et al., 2006]. An audio track is usually partitioned into non-overlapping audio segments multiple seconds in length, and segment-wise metrics are then averaged over each audio track or the whole dataset to evaluate model performance. Following the procedure used for the SiSec separation campaign [Rafii et al., 2017], these segments are one second long.

Issues with current evaluation metrics The SDR computation is problematic when the true source is silent or near-silent. In case of silence, the SDR is undefined ($\log(0)$), which happens often for vocal tracks. Such segments are excluded from the results, so performance on these segments is ignored. For near-silent parts, the SDR is typically very low when the separator output is quiet, but not silent, although such an output is arguably not a grave error perceptually. These outliers are visualised using model M5 in Figure 5.3. Since

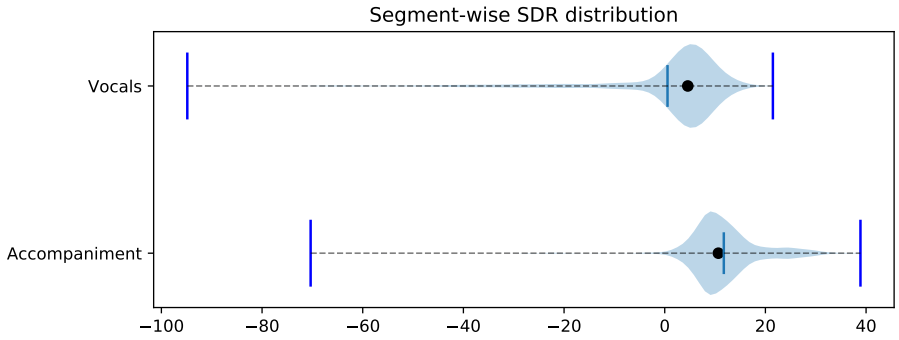


Figure 5.3: Violin plot of the segment-wise SDR values on the MUSDB test set for model M5. Black points show medians, dark blue lines the means.

the mean over segments is usually used to obtain overall performance measures, these outliers greatly affect evaluation results.

Since the collection of segment-wise vocal SDR values across the dataset is not normally distributed (compare Figure 5.3 for vocals), the mean and standard deviation are not sufficient to adequately summarise it. As a workaround, we take the median over segments, as it is robust against outliers and intuitively describes the minimum performance that is achieved 50% of the time. To describe the spread of the distribution, we use the median absolute deviation (MAD) as a rank-based equivalent to the standard deviation (SD). It is defined as the median of the absolute deviations from the overall median and is easily interpretable, since a value of x means that 50% of values have an absolute difference from the median that is lower than x.

We also note that increasing the duration of segments beyond one second alleviates this issue by removing many, but not all outliers. This is more memory-intensive and presumably still punishes errors during silent sections most.

Model comparison Table 5.2 shows the evaluation results for singing voice separation. The low vocal SDR means and high medians for all models again demonstrate the outlier problem discussed earlier. The difference output layer does not noticeably change performance, as model M2 appears to be only very slightly better than model M1. Initial experiments without fine-tuning showed a larger difference, which may indicate that a finer adjustment of weights makes constrained outputs less important, but they could still enable the usage of faster learning rates. Introducing context noticeably improves performance, as model M3 shows, likely due to better predictions at output borders. The stereo modeling in model M4 yields improvements especially for accompaniment, which

		M1	M2	M3	M4	M5	M7	U7	U7a
Voc.	Med.	3.90	3.92	3.96	4.46	**4.58**	**3.49**	2.76	2.74
	MAD	3.04	3.01	3.00	3.21	3.28	2.71	2.46	2.54
	Mean	-0.12	0.05	0.31	**0.65**	0.55	-0.23	-0.66	**0.51**
	SD	14.00	13.63	13.25	13.67	13.84	13.00	12.38	10.82
Acc.	Med.	7.45	7.46	7.53	**10.69**	10.66	**7.12**	6.76	6.68
	MAD	2.08	2.10	2.11	3.15	3.10	2.04	2.00	2.04
	Mean	7.62	7.68	7.66	**11.85**	11.74	**7.15**	6.90	6.85
	SD	3.93	3.84	3.90	7.03	7.05	4.10	3.67	3.60

Table 5.2: Test set performance metrics (SDR statistics, in dB) for each singing voice separation model. Best performances overall and among comparison models are shown in bold.

Vocals				Other			
Med.	MAD	Mean	SD	Med.	MAD	Mean	SD
3.0	2.76	-2.10	15.41	2.03	1.64	1.68	6.14
Bass				Drums			
Med.	MAD	Mean	SD	Med.	MAD	Mean	SD
2.91	2.47	-0.30	13.50	4.15	1.99	2.88	7.68

Table 5.3: Test performance metrics (SDR statistics, in dB) for our multi-instrument model M6

may be because its sounds are panned more to the left or right channels than vocals. The learned upsampling (M5) slightly improves the median, but slightly decreases the mean vocal SDR. The small differences could be explained by the low number of weights in learned upsampling layers. For multi-instrument separation, we achieve slightly lower but moderate performance (M6), as shown in Table 5.3, in part due to less training data.

U7 performs worse than our comparison model M7, suggesting that our network architecture compares favourably to the state-of-the-art architecture since all else is kept constant during the experiments. However, U7 stopped improving on the training set unexpectedly early, perhaps because it was not designed for minimising an audio-based MSE loss or because of effects related to backpropagating gradients through the inverse STFT. In contrast, U7a showed expected training behaviour using the magnitude-based loss. Our model also outperforms U7a, yielding considerably higher mean and median SDR scores. The mean vocal SDR is the only exception, arising since our model has more outlier segments, but better output the majority of the time.

Models M4 and M6 were submitted as STL1 and STL2 to the SiSec cam-

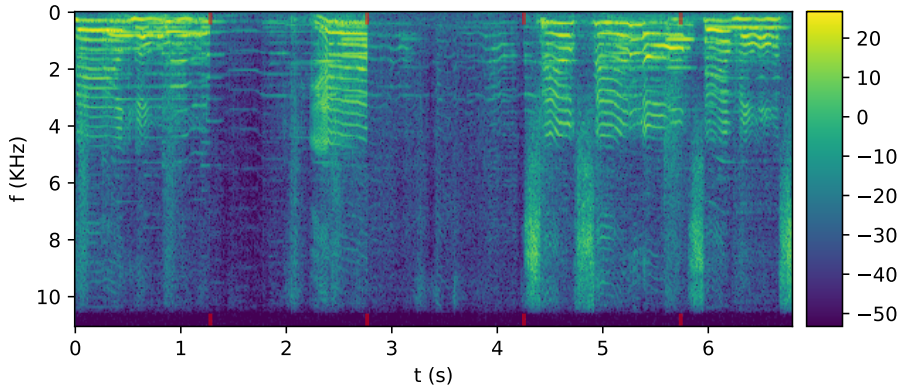


Figure 5.4: Power spectrogram (dB) of a vocal estimate excerpt generated by a model without additional input context. Red markers show boundaries between independent segment-wise predictions.

paign [Stöter et al., 2018]. For vocals, M4 performs better or as well as almost all other systems. Although it is significantly outperformed by submissions UHL3, TAK1-3 and TAU1, all of these except TAK1 used an additional 800 songs for training and thus have a large advantage. M4 also separates accompaniment well, although slightly less so than the vocals. We refer to [Stöter et al., 2018] for more details.

5.2.3.2 Qualitative results and observations

As an example of problems occurring when not using a proper temporal context, we generated a vocal source estimate for a song with the baseline model M1, and visualised an excerpt using a spectrogram in Figure 5.4. Since the model's input and output are of equal length and the total output is created by concatenating predictions for non-overlapping consecutive audio segments, inconsistencies emerge at the borders shown in red: the loudness abruptly decreases at 1.2 seconds, and a beginning vocal melisma is suddenly cut off at 2.8 seconds, leaving only quiet noise, before the vocals reappear at 4.2 seconds. A vocal melisma with only the vowel "a" can sound similar to a non-vocal instrument and presumably was mistaken for one because no further temporal context was available. This indicates that models without additional input context suffer not only from inconsistencies at such segment borders, but are also less capable of performing separation there whenever information from the context is required.

Larger input and output sizes alleviate the issue somewhat, but the problems at the borders remain. Blending the predictions for overlapping segments [Grais et al., 2018] is an ad-hoc solution, since the average of multiple predicted audio signals might not be a realistic prediction itself. For example, two sinusoids

with equal amplitude and frequency, but opposite phase would cancel each other out. Blending should thus be avoided in favour of our context-aware prediction framework.

5.2.4 Discussion and conclusion

We proposed the Wave-U-Net for end-to-end audio source separation without any pre- or postprocessing, and applied it to singing voice and multi-instrument separation. A long temporal context is processed by repeated downsampling and convolution of feature maps to combine high- and low-level features at different time-scales. As indicated by our experiments, it outperforms the state-of-the-art spectrogram-based U-Net architecture [Jansson et al., 2017] when trained under comparable settings. Since our data is quite limited in size however, it would be interesting to train our model on datasets comparable in size to the one used by [Jansson et al., 2017] to better assess respective advantages and disadvantages.

We highlight the lack of a proper temporal input context in recent separation and enhancement models, which can hurt performance and create artifacts, and propose a simple change to the padding of convolutions as a solution. Similarly, artifacts resulting from upsampling by zero-padding as part of strided transposed convolutions can be addressed with a linear upsampling with a fixed or learned weight to avoid periodic artifacts.

Finally, we identify a problem in current SDR-based evaluation frameworks that produces outliers for quiet parts of sources and propose additionally reporting rank-based metrics as a simple workaround. However, the underlying problem of perceptual evaluation of sound separation results using SDR metrics still remains and should be tackled at its root in the future.

To also exploit the benefits of the Wave-U-Net approach in sequence modelling tasks, the network architecture needs to be adapted so it does not access future samples to predict the output at a given time-step (that is, satisfying the auto-regressive condition). In the following Section, we will demonstrate how the Wave-U-Net can be naturally adapted for this setting, giving rise to the "Seq-U-Net", which we experimentally compare in a variety of contexts such as text and raw audio generation.

5.3 Seq-U-Net

Sequence modelling is an important problem central to many application domains, including language, audio, and video generation [Bai et al., 2018, Yu et al., 2017, Trinh et al., 2018]. In some of these applications, the sequences can be millions of time-steps in length (e.g. in the case of audio generation due to the high sampling

rate of audio signals), and it can be vital to model the long-term dependencies present in such sequences (for example to be able to repeat a melody in a music piece that occurred a minute earlier).

This problem is often framed as the task of predicting the next element in a sequence given all of the elements observed so far, giving rise to auto-regressive models. Recurrent neural networks (RNNs) are often used in this context since they can theoretically remember inputs for an arbitrary number of time-steps, and also offer quick inference at test time as the hidden state carries all the information about previous sequence elements and only needs to be updated using the next element. However, in practice, these models can be difficult [Bengio et al., 1994] and slow [Trinh et al., 2018] to train due to their strictly sequential nature. More recently, CNNs with dilated filters were shown to be competitive approaches for sequence modelling. Instead of relying on recurrence to retain information over a large number of steps, which might be difficult to achieve in practice, CNNs such as the temporal convolutional network (TCN) [Bai et al., 2018] and Wavenet [van den Oord et al., 2016] access far-away time-steps more directly through their dilated filters.

Despite their impressive performance, these architectures suffer from two issues. Firstly, each convolutional layer operates at the same time resolution as the input. This results in a high memory usage and training time especially with long sequences, rendering long-term modelling infeasible even with large scale, multi-GPU training [van den Oord et al., 2016]. Secondly, inference is slow as elements have to be predicted sequentially and require a forward pass through the CNN's many layers. Although re-using layer outputs from previous steps helps, all layers still have to be traversed and updated to predict the next sequence element.

In this context, the "slow feature analysis" [Wiskott and Sejnowski, 2002] hypothesis states that for a wide variety of tasks, important features of an input signal vary only slowly over time. This leads to an interesting approach of increasing efficiency by computing some features at lower sampling rates compared to the input without compromising model performance. Notably, U-Nets [Ronneberger et al., 2015] already incorporate the equivalent of this principle for image processing, by computing features at different time-scales with two-dimensional convolutions and combining them to make predictions at the same resolution as the input. We base our model on the Wave-U-Net described in Section 5.2, as it should be able to process many kinds of temporal sequences, not just audio signals. We show how to adapt it for our auto-regressive setting by making all convolutions causal, such that each prediction for the next time-step can only depend on past inputs.

As a result, we obtain the "Seq-U-Net"[3], a general-purpose network architecture that is not limited to audio tasks but can be applied to a wide range of sequence modelling problems – while providing considerable efficiency improvements over TCN and Wav enet. Inference is greatly accelerated by only computing new layer activations if they are not decimated in the downsampling process. This time-variant processing gives each layer its own "update rate", which is in contrast to fully-convolutional TCN and Wavenet approaches. In particular, we compare to TCN in the context of word- and character-level language modelling and symbolic music generation. Additionally, we tackle the task of generating piano music directly in the time-domain and compare performance with a Wavenet reimplementation using a log-likelihood metric as well as listening tests. Overall, we find that our architecture achieves competitive results while requiring less memory and training time.

5.3.1 Method

We present two variants of our multi-scale approach. The first is an adaptation of the Wave-U-Net to the auto-regressive setting and shown in Section 5.3.1.1. The second variant, presented in Section 5.3.1.2, further adds residual connections to stabilise training for tasks with very long-term dependencies such as raw audio generation.

5.3.1.1 Seq-U-Net

Our model is based on the Wave-U-Net presented in Section 5.2 and shown in Figure 5.5. The network features L *levels* of downsampling (DS) and upsampling (US) blocks, and a convolutional bottleneck and output layer. Each downsampling block features a convolution, whose outputs are used as a shortcut connection for the respective upsampling block, followed by another convolution with stride k to downsample the features across time. Each upsampling block has a transposed convolution with stride k to upsample the previously obtained coarse-grained features. The result is concatenated with the features from the shortcut connection, and input to another convolution to combine high- and low-level features. We set the stride k to 2 for our experiments. All convolutions have the same filter width and a LeakyReLU activation followed by Dropout, except for the output convolution.

Like in the original Wave-U-Net, the convolutional layers do not use zero-padding so that all model predictions are made with the necessary input context. As a result, there are more feature frames in the shortcut output of a DS block than in the output of the transposed strided convolution in the corresponding

[3]Code available at `https://github.com/f90/Seq-U-Net`

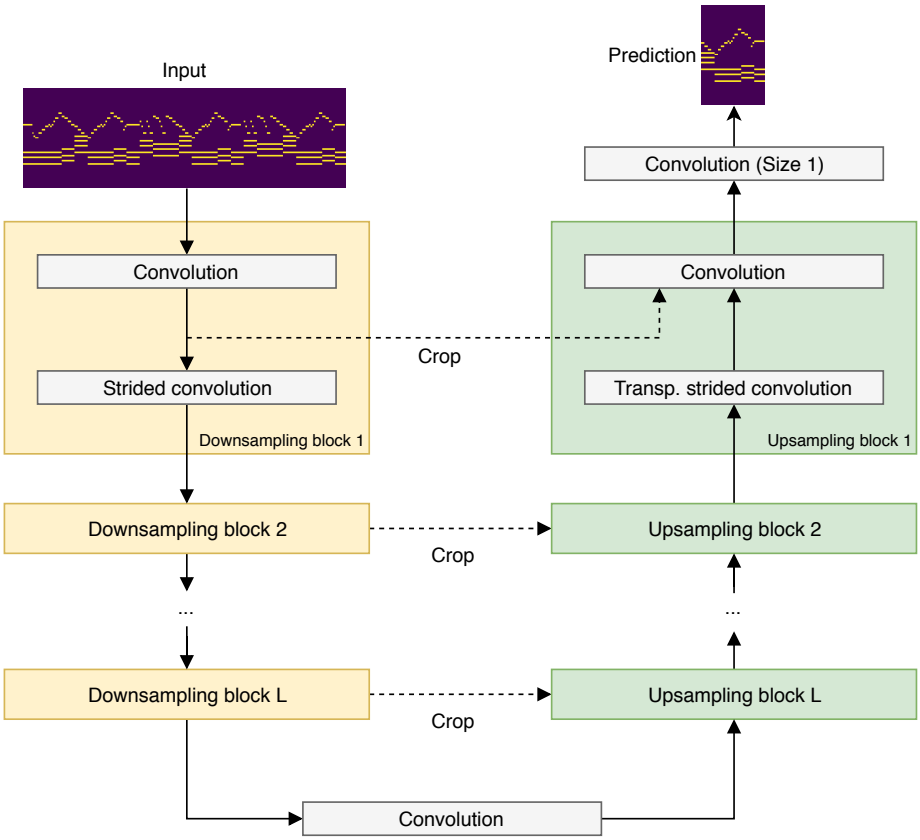


Figure 5.5: Architecture diagram of our proposed model. 1D Convolutions are applied across time with LeakyReLU activations followed by Dropout. Strided and transposed strided convolutions are used for down- and upsampling the features, respectively. Since the convolutions do not use padding, the output is smaller than the input and skip connections need to be cropped at the front.

US block. Zeros are prepended to the beginning of input sequences to allow predicting the first sequence elements. In the Wave-U-Net, the outputs at each level of the network are interpreted as features describing the center part of the input, so the shortcut features are center-cropped before concatenation. Consequently, source signals are predicted for the center part of the mixture excerpt.

Our key idea is to interpret the filters as causal instead: the output of a filter covering input timesteps $n-k$ to $n+k$ should now help predict input x_{n+k+1} *outside* of its receptive field instead of some feature of the input at timestep n, i.e. the current source audio signal. Therefore, we instead crop the first feature frames of each shortcut connection to make sure that features are aligned in time properly. As a result, we obtain an auto-regressive model for sequence modelling, similar to Wavenet and TCN, but significantly sparser in terms of activations

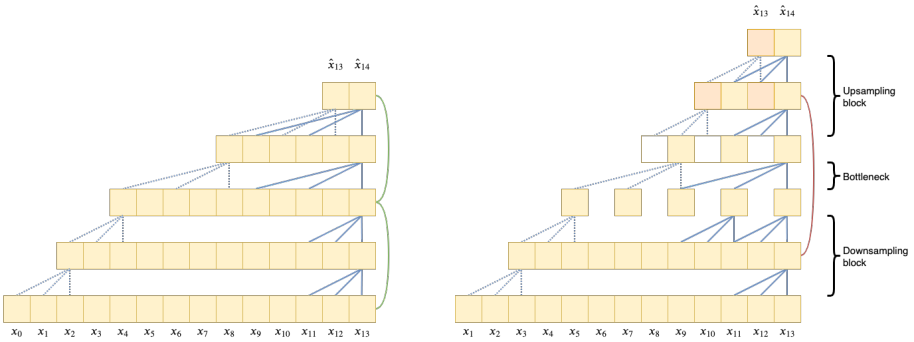


Figure 5.6: Compared to TCN (left, two residual blocks), Seq-U-Net (right, one down- and upsampling block) computes features only at certain intervals to save memory and training time. Zero-padding is used in the upsampling blocks (white squares), leading to different computational paths throughout the network (red squares). The red line indicates feature cropping and concatenation.

due to the decreased resolution in most of the layers.

Fast inference From a signal processing perspective, TCN and Wavenet are time-invariant systems as they apply the same set of operations at each time step. Time-invariant processing, however, is not required in autoregressive models. In contrast, the multi-scale architecture of Seq-U-Net allows us to employ a time-variant processing scheme (inspired by Koutník et al. [2014]) that drastically accelerates inference, as many operations do not have to be computed at every step: If an output computed for the latest time-step in a DS block is decimated, only the US blocks on the same or higher resolution need to be updated, since the input to the other blocks does not change. This means that a block on level $i \in \{1, \ldots, L\}$ only needs to be updated every k^{i-1} time-steps. To implement this procedure, all blocks are given an internal clock based on their level to determine when to compute a new output. To predict the very first sample from a given context, a normal forward-pass is conducted and caches for the resulting layer activations are set up before switching to the above step-wise procedure.

5.3.1.2 Residual variant

Since raw audio generation benefits from a large receptive field, we employ much deeper instances of our model for the experiment in Section 5.3.3.2. With this increase in layers however, we observed training instability. Residual networks can be trained stably even with hundreds of layers [He et al., 2015], so we also propose a residual variant of our model.

Compared to the baseline model from Section 5.3.1.1, we employ an additional convolution on the input with F output channels, and also use F input and output

channels for all up- and downsampling blocks to allow for residual connections. We replace each convolutional layer in the base model with a residual layer similar to the one in Wavenet [van den Oord et al., 2016], whose outputs y are given by

$$y = I(x) + \tanh(C_1(x)) \cdot \sigma(C_2(x)), \tag{5.2}$$

where x are the layer inputs, σ is the sigmoid function, C_i applies convolutional layer i to its input and I processes the input x to provide an identity connection in case the convolutions change the feature dimensionality. Dropout was omitted in this variant since overfitting was not a large concern in our audio generation experiments, but could readily be added to the residual convolutions.

For the convolutions with stride used in the DS blocks, I first decimates the input x to provide the identity for the residual layer. For the transposed convolutions with stride in the US blocks, I takes the input and repeats the feature vector at each time step $k-1$ times to perform upsampling[4]. For both down- and upsampling, I finally crops the resulting feature sequence at the front to ensure it matches the number of residual features, which is reduced due to not using padding for convolutions. To refine the high-resolution shortcut features using the low-resolution features from the upsampling path, we use the shortcut as input x and use the concatenation of the shortcut and the upsampled features as input to the residual convolutions C_i.

To easily scale the network in size for more complex tasks, we employ $D+1$ residual layers in each block (one layer for up- or downsampling), with D as hyper-parameter, allowing features to be processed more flexibly at each time resolution.

5.3.2 Complexity analysis

We will analyse the memory consumption and computational complexity of our approach at both training and test time and compare with Wavenet[5].

5.3.2.1 Training

Due to the size N of the receptive field increasing exponentially with the number of layers for the Seq-U-Net and Wavenet, roughly $L = \log_k(N)$ levels of processing are required. For the Wavenet, we define k as the factor with which dilation increases in each layer.

When presented with $I \geq N$ inputs during training, Wavenet needs to compute I feature activations in each of the L layers, since it operates on the

[4]This operation does not violate the auto-regressive condition.

[5]Comparison with TCN is omitted as it is very similar to Wavenet but differs slightly in the number of layers per level of resolution

same resolution as the input, reaching a total of $I \cdot \log_k(N)$. The Seq-U-Net on the other hand computes $3I + \frac{I}{k}$ feature activations in the first down- and upsampling block, $3\frac{I}{k} + \frac{I}{k^2}$ on the second level, and so on, in addition to a bottleneck convolution with $\frac{I}{k^L}$ outputs. For the Seq-U-Net, we thus obtain at most $\sum_{i=0}^{L} 4\frac{I}{k^i} \leq 8I$ feature activations regardless of the number of layers[6]. The above calculation not only demonstrates the time complexity, but also the required memory, since the computed feature activations need to be maintained for the backward-pass.

5.3.2.2 Inference

At test time, auto-regressive models such as Wavenet and Seq-U-Net require a forward pass to generate the next element in the sequence, which can be prohibitively slow when sequences are long (e.g. in audio generation) or when a real-time application is desired. While caching previously computed outputs in the Wavenet reduces computation time, it still involves evaluating all L layers, which especially affects deep models (e.g. $L = 30$ by Kalchbrenner et al. [2018]).

In the Seq-U-Net, each level in the network only has to be updated at certain intervals as described in Section 5.3.2. In particular, the average number of levels we have to update for each time-step is $\sum_{i=1}^{L} \frac{1}{k^{i-1}} \leq 2$ and thus a constant number of layers independent of the number of levels L in the network. While this is an amortised analysis of the average time per step, in the worst case all layers need to be updated, although this is not relevant for offline sequence generation.

5.3.3 Experiments and Results

We evaluate our method on a variety of sequence modelling tasks regarding its performance, training time and memory complexity. Due to the architectural similarity, we will firstly compare our method with TCN in Section 5.3.3.1 on language modelling as well as symbolic music modelling. To test whether our model can capture long-term dependencies, we also compare to TCN on a synthetic copy task and to a Wavenet baseline on the task of audio generation in the time-domain. Note that the Wave-U-Net can not be used as a baseline model for these experiments, since it has access to sequence element x_{t+1} when predicting the successor to x_t and can therefore easily achieve perfect prediction.

For time and memory measurements, we use a single NVIDIA GTX 1080 GPU with Pytorch 1.2, CUDA 9 and cuDNN 7.5[7]. We compare the average time

[6]This disregards the reduction in size due to not using padding for convolutions since it occurs in all models

[7]We use Pytorch's benchmark mode to find the best algorithm for training each network.

required for each training step[8] and the maximum memory allocated throughout a training epoch[9].

5.3.3.1 Comparison with TCN

We will compare our model against TCN across three sequence modelling tasks. To match model complexity, we use the same filter length, Dropout rate, and levels of resolution, which results in very similar receptive field size. Then, the number of features in each layer is adapted for Seq-U-Net so it matches TCN in the number of parameters.

We optimise each model for 100 epochs using a batch size of 16 and an Adam optimiser with initial learning rate α, which is reduced by half if validation performance did not improve after P epochs and more than 10 epochs have passed since the beginning of training. Finally, the model that performed best on the validation set is selected.

To prevent the training procedure from favouring one model over the other, we perform a hyper-parameter optimisation over the learning rate $\alpha \in [e^{-12}, e^{-2}]$ and optional gradient clipping with magnitudes between $[0.01, 1.0]$. This hyper-parameter optimisation is performed for each combination of model and task using a tree of Parzen estimators[10] to find the minimum validation loss. All hyper-parameters are shown in Table 5.6.

Character-based language modelling We perform character-based language modelling, where the task is to predict the next character given a history of previously observed ones, on the PTB dataset [Marcus et al., 1993]. The average cross-entropy loss is used as training objective, and patience is set to $P = 5$.

For both models, we use 100-dimensional character embeddings with 0.1 Dropout as input, and their output is projected back to character probabilities using the transposed version of the embedding matrix. We evaluate models using the bits-per-character (bpc) metric.

As shown in Table 5.4, our model performs as well as its TCN counterpart in this regard, while requiring 59% less time per training step, and 32% less GPU memory during training. These results suggest that many of the required features are on a higher level of abstraction and vary only slowly, e.g. per word or per sentence, and so do not need to be recomputed for each new character – a hypothesis also put forth by Chung et al. [2016].

[8] Extra time due to data loading is not included
[9] Does not include memory used for purposes such as caching
[10] "Hyperopt" package: `http://hyperopt.github.io/hyperopt/`

Task	Model	Train	Test	Time (s)	Memory
Char-LM	TCN	1.066	1.31	0.0694	445.9
Char-LM	Seq-U-Net	1.08	1.30	0.0286	304.9
Word-LM	TCN	47.21	108.47	0.0480	580.5
Word-LM	Seq-U-Net	40.43	107.95	0.0234	382.1
M-Muse	TCN	5.789	6.931	0.0059	108.5
M-Muse	Seq-U-Net	5.794	6.969	0.0065	75.3
M-Nott	TCN	1.409	2.783	0.0071	73.1
M-Nott	Seq-U-Net	1.850	2.97	0.0067	52.5
M-JSB	TCN	6.178	8.154	0.0034	13.1
M-JSB	Seq-U-Net	6.151	8.173	0.0037	8.2
Piano	Wavenet	1.76	1.88	1.4616*	5294*
Piano	Seq-U-Net**	1.83	1.93	0.3621*	1514*

* Measurements were taken with a batch size of 2 instead of 16 due to the high amount of memory required.
** Residual variant

Table 5.4: Performance for Seq-U-Net (lower is better) and comparison models across different tasks ("M-" indicates symbolic music modelling). Times denote the duration for a forward- and backward-pass, averaged over a whole epoch. "Memory" indicates GPU memory consumption in MB

Word-based language modelling For our second experiment, we perform word-based language modelling, which involves predicting the next word following a given sequence of words. As in the previous experiment, we use the PTB dataset with a vocabulary of 10,000 words. Following TCN's experimental set-up [Bai et al., 2018], we use 600-dimensional word embeddings with 0.25 Dropout as input, and use the transpose of the embedding matrix to project the 600-dimensional outputs from the models to probability vectors over all words. For training, we minimise the average cross-entropy with a patience of $P = 5$, and for evaluation we use the per-word perplexity.

Similarly to the results for character-based language modelling in Section 5.3.3.1, Table 5.4 shows that both models perform very similarly, but the Seq-U-Net architecture is substantially more efficient to train (reducing the training time by 51% and memory usage by 34%).

Symbolic music modelling For our final comparison with TCN on real-world data, we model polyphonic music in the symbolic domain. Each music piece is represented as a piano roll – a binary matrix of size $88 \times T$ that indicates which of the 88 pitches are active at each of the T time frames. For simplicity, we assume that pitch activations at a given time frame are independent of each other[11]. Our

[11] Common in music transcription, e.g. Ycart and Benetos [2018]

models predicts a whole time-frame at each step in an auto-regressive manner, and we use the sum of binary cross-entropies over each pitch, averaged over all time frames as training objective. We use a patience of $P = 10$ for early stopping.

Three different datasets of varying complexity and content are used: Muse[12], Nottingham (Nott)[13] and the JSB chorales [Allan and Williams, 2005]. For evaluation, we use the frame-wise perplexity introduced by Boulanger-Lewandowski et al. [2012].

Table 5.4 shows the perplexity on the training and test sets for both models on all datasets. We find that both models are very closely matched in terms of training and test perplexity on the Muse and JSB datasets. For the Nott dataset, TCN achieves a noticeably lower perplexity than the Seq-U-Net on the training partition. This performance gap also appears on the test set, although it is considerably smaller, indicating that incorporating the slow feature hypothesis induces a regularising effect on the model.

For these datasets, no improvement in training time is observed, unlike the previous language modelling experiments. This is due to the much smaller size of the models, where the higher number of convolutional layers in the Seq-U-Net has a larger impact than the reduction in computation time for each layer. Nevertheless, the memory footprint is substantially reduced by an average of 32%.

Copy task Finally, we compared our model to TCN on the *copy task*, following the experimental setup outlined by Bai et al. [2018]. The input to the model is a one-dimensional sequence consisting of 10 integers randomly chosen between 1 and 8, followed by M zeroes, and 11 entries filled with the digit 9, acting as a signal for the model to output the initial 10-number sequence at the end of the input sequence.

Using the same setting of $M = 1000$ used by Bai et al. [2018], we found that Seq-U-Net was not able to retain the number sequence and output it at the end (reaching an accuracy of 12.7%), in contrast to TCN. Theoretically, this can be explained by the resampling operations contained in the Seq-U-Net, through which the number sequence needs to be transported. Neighbouring elements (feature vectors) of the sequence need to be encoded into a single feature vector so that subsequent downsampling of this sequence of feature vectors does not result in information loss. Similarly, even if the information successfully passes through all downsampling layers, the original sequence has to be decoded in the upsampling path. Both of these operations would require very specific

[12] See `http://www-etud.iro.umontreal.ca/~boulanni/icml2012`
[13] See `http://ifdo.ca/~seymour/nottingham/nottingham.html`

Model	Layers	Features	Context	Filter width
Wavenet	13	128	32764	2
Seq-U-Net	11	180	32748	5

Table 5.5: Models used for audio generation. Context is given as a number of audio samples.

configurations of the convolutional filters to be successful. In other words, the Seq-U-Net is built to retain low-frequency features over a large amount of steps, but the number sequence represents high-frequency information. However, it seems that retaining such high-frequency information over large numbers of time-steps is rarely needed in many real-world applications, since Seq-U-Net performs well on all real-world benchmarks presented here.

5.3.3.2 Raw audio generation

To test whether our model can capture long-term dependencies found in complex real-world sequences, we apply it to the generation of audio waveforms, using the residual variant presented in Section 5.3.1.2. Since our architecture resembles the Wavenet with its use of stacks of residual convolutions, we use it as our comparison model in the following.

In particular, we use the classical piano recordings as used by Dieleman et al. [2018] amounting to about 607 hours in duration, and partition them into a training and test set, while avoiding pieces overlapping between the two partitions. Note that our version of the dataset is different as we were not able to obtain all the recordings listed by Dieleman et al. [2018].

We train two models in this experiment, listed in Table 5.5. The first one is a Wavenet baseline comprised of 4 Wavenet stacks with 13 dilated convolutional layers each and 512 features in the skip connection, and the second one is a Seq-U-Net model that matches the Wavenet in terms of receptive field size, and uses a residual depth of $D = 2$.

Besides downsampling the audio to 16 KHz mono signals, no further pre-processing is applied. During training, audio excerpts are loaded from random positions within the audio files, and each audio sample is transformed into a 256-dimensional one-hot vector using 8-bit mu-law encoding, following the Wavenet approach [van den Oord et al., 2016]. A training batch consists of 16 examples and uses the last 5000 audio samples in each example as simultaneous training targets for the model. The average cross-entropy is minimised over 246000 iterations (equivalent to just over one epoch) with an Adam optimiser and a learning rate of 0.0005.

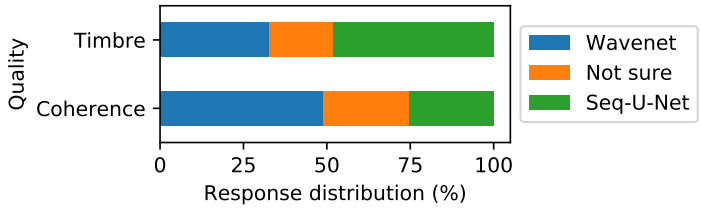


Figure 5.7: Results of the listening test, showing the overall distribution of responses for both the timbre and the musical coherence questions

Experimental setup For evaluation, we report the likelihood of the models in bits per audio samples (bpa) on the test set. However, the bpa metric might not reflect perceptual audio quality very well, especially since the model uses its own predictions as input and not real samples at test time. This discrepancy is well known in the literature [Huszár, 2015], and we also found in practice that the two models vary in their stability at generation time. While the Wavenet starts to introduce progressively more noise into its outputs with longer generation, the Seq-U-Net appears stable throughout. Since this effect is very pronounced with durations of 10 seconds or longer, making Seq-U-Net clearly preferable, we conducted a listening test with samples of 5 seconds. We used a temperature of 0.95, meaning the unnormalised model outputs were divided by 0.95 before applying the softmax to obtain probabilities. In preliminary experiments, we found this stabilises the generation process, resulting in increased quality for both models.

Each of the 20 questions presented the participant with a 1.5 second excerpt of real piano randomly sampled from our test dataset. This was followed by two continuations produced by our two models that also include the real excerpt in the beginning. This conditional generation setting allows directly comparing between outputs of different models for the same input context: The participants were asked which excerpt has "better timbre (does it sound like a piano, is the audio free of distortions?)" and "more musical coherence (with respect to melody, harmony, rhythm)". An additional "Not sure" option was available when the participant thinks the quality is the same for both excerpts. The total number of participants is 22, with the majority being between the ages of 25 and 40. We did not select for specific levels of musical training.

Results As seen in Table 5.4, the Wavenet slightly outperforms the Seq-U-Net in terms of the bpa metric, albeit achieving a small relative improvement of 2.6% on the test set, indicating the models are closely matched in terms of performance. The training set results indicate this might be due to the Wavenet

Task	Model	W	L	H	Drop.	Context	Params	P	LR	Clip
Char-LM	TCN	3	4	600	0.1	80	5.9M	5	0.00014	0.213
Char-LM	Seq-U-Net	3	4	390	0.1	73	5.9M	5	0.00073	No
Word-LM	TCN	3	4	600	0.5	73	14.7M	5	0.00115	No
Word-LM	Seq-U-Net	3	4	390	0.5	73	14.9M	5	0.00037	0.722
M-Muse	TCN	5	4	215	0.2	Full	1.7M	10	0.00023	No
M-Muse	Seq-U-Net	5	4	150	0.2	Full	1.7M	10	0.00047	No
M-Nott	TCN	5	4	215	0.2	Full	1.7M	10	0.000067	0.601
M-Nott	Seq-U-Net	5	4	150	0.2	Full	1.7M	10	0.00108	No
M-JSB	TCN	3	2	220	0.5	Full	534k	10	0.00134	No
M-JSB	Seq-U-Net	3	2	170	0.5	Full	522k	10	0.00051	0.324

Table 5.6: Hyper-parameters used for TCN and Seq-U-Net comparisons. W the convolutional filter width, L the number of layers, H the number of filters in each convolutional layer and P is the patience for early stopping. LR and Clip are the best learning rate and clipping magnitude found through hyper-parameter optimisation.

fitting the training set more closely in the given number of training iterations. At the same time, the required training time and memory are drastically reduced for the Seq-U-Net by factors of 4 and 3.5, respectively.

The results of the listening test are shown in Figure 5.7. While the Seq-U-Net exhibits better timbral characteristics than the Wavenet, it falls behind in terms of musical coherence. We suspect this is due to the Seq-U-Net sometimes producing an unexpected transition from the real excerpt to the generated section, but then producing sounds more stably as time goes on. Overall, the two models appear to have different strengths and weaknesses. Additionally, the high amount of "Not sure" responses, especially for such a sensitive paired discrimination task, indicates that the models are quite evenly matched in this setting.

Finally, we measure the performance impact of our inference method introduced in Section 5.3.1.1 by comparing to the Wavenet's generation speed when caching previous activations. With a batch size of 1 on a single NVIDIA GTX 1080 GPU, we achieve 69 audio samples per second for the Wavenet, and 309 for the Seq-U-Net and thereby a speed-up with a factor greater than 4.

5.3.4 Discussion and conclusion

The predictive performance of the Seq-U-Net as outlined in Table 5.4 is remarkably similar to that of Wavenet and TCN comparison models across all tasks we tested. While efficiency gains are not very noticeable for very small instances of our model with few levels of resolution, they rapidly increase when moving towards larger and deeper models as used in language and audio modelling, and

we can expect these gains to become more pronounced for even deeper models with even longer receptive fields.

Since the metrics used in Table 5.4 are based on how much probability the models assign to the test data (log-likelihood) and not directly on how realistic their generated output is, we performed a listening test for the piano audio generation task. Surprisingly, despite better log-likelihood, our implementation of the Wavenet accumulates noise during generation, making it unsuitable to generate longer music pieces, whereas the Seq-U-Net is stable but less capable of smoothly continuing the real excerpts, for reasons that remain unclear. A more unified approach to training and evaluating generative models would be desirable, so models can be more directly adapted for stability during generation time, instead of relying on architecture choices alone to ensure stability.

We demonstrated how a causal variant of a U-Net architecture with one-dimensional convolutions across the time domain can perform on par with existing state-of-the-art models in a variety of real-world sequence modelling tasks, while significantly reducing training time and memory requirements. Leveraging the idea that many relevant features in real-world sequences are only slowly varying over time allows the use of convolutional layers that compute features at progressively lower resolutions. These efficiency gains make it feasible to train generative models with much longer receptive fields in the future, which can be very useful in domains such as music and language generation. While results on the synthetic copy task show that high-frequency information can not be retained over large numbers of time steps, the competitive performance of our model on real-world benchmarks suggests that only modelling long-term dependencies between "slow features" might be sufficient – although this should be investigated further in the future.

A limitation of our approach is that the levels of resolution along with the processing capacity at each resolution has to be manually pre-defined, which could limit performance. Future work could include potential solutions as used in the Phased LSTM [Neil et al., 2016] so the model can adapt its levels of resolution more dynamically to the task.

Finally, attention mechanisms have shown great potential for sequence modelling and could be integrated into our approach by using attention operations in each down- and upsampling block alongside or instead of convolutions to further improve performance, as suggested by Child et al. [2019].

5.4 Conclusion

In this Section, we developed end-to-end models for processing high-dimensional input sequences. A key challenge for these models is computational efficiency,

as sequences are very high-dimensional in some domains, such as audio signals encountered in machine listening. Our solution is based on the slow feature hypothesis – we use a CNN architecture where different layers operate on different resolutions, creating a feature hierarchy where many layers output features at a much lower resolution than that of the input, saving memory and computation time.

This approach can be applied to non-causal ("offline") prediction, where the whole input sequence can be used to predict the output sequence, giving rise to Wave-U-Net as described in Section 5.2. It can also be straightforwardly extended to the causal ("online") setting, where elements in the output sequence are predicted based only on the input elements observed so far. The resulting Seq-U-Net presented in Section 5.3 can be used for auto-regressive generative modelling, such as music generation using the raw waveform.

These end-to-end models can be combined with powerful, data-driven approaches to incorporate model priors, some of which we proposed in Sections 3 and 4, to obtain good generalisation even when not much labelled data or explicit prior knowledge is available. Additionally, when large amounts of labelled data are available, our proposed models can allow for better scalability and performance for machine listening tasks since they can flexibly learn feature representations directly from raw audio. In contrast, current machine listening models make use of fixed spectrogram representations and reductive features such as Mel-frequency cepstral or chroma coefficients that risk discarding a lot of relevant information in the input, thereby limiting performance.

Chapter 6

Conclusions and further work

6.1 Summary of contributions

The goal in this thesis is to build machine listening models that generalise well on tasks with only a small amount of annotated data. To this end, we developed approaches that add powerful, data-driven regularisation terms to the model's training objective and that make use of additional related datasets, which are often available in practice.

In Chapter 3, we developed a novel framework based on adversarial learning that allows incorporating additional types of datasets into training. These datasets may contain only a subset of all features used by the model. We applied this technique in the context of image-to-image translation as well as audio source separation, where additional datasets comprised of individual input or output samples are often available that can not be used in a traditional supervised learning framework. In our experiments, this leads to performance increases over baseline models that are not able to make use of the additional data. Furthermore, we retain the adversarial learning framework, as our adaptation subdivides the discriminator network so that overall it estimates the same density ratio as the discriminator in the original GAN [Goodfellow et al., 2014], allowing us to keep generator training unchanged.

As our second main contribution, we investigated methods to exploit the similarity between machine listening tasks in Chapter 4, focusing on music information retrieval tasks in particular. In an initial study in Section 4.2, we found that employing a simple multi-task learning approach to simultaneously detect and separate the singing voice in music mixtures results in better performance

than training separate models for each of the two tasks. Expanding on this first experiment that involved only two tasks, we present a meta-learning approach in Section 4.3 that aims to find a parameter initialisation of a base network that leads to good performance on a given task after a small amount of task-specific training, for which we use a total of ten MIR tasks. While model performance sometimes degrades compared to simply training the model from scratch on the task of interest, in other cases we observe an impressive performance increase, suggesting that the shared structure between MIR tasks is encoded into the initial parameters so it can be successfully exploited by the model. This this initial exploration is particularly valuable as it can inform future work in this under-explored area, for example with regards to the steps needed to obtain more consistent performance improvements.

Finally, we help pave the way towards more end-to-end machine listening models that can be flexibly combined with data-driven model priors such as the ones presented in this thesis, thereby avoiding cumbersome and non-scalable feature preprocessing. In particular, we presented two CNNs – the non-causal "Wave-U-Net" model that we successfully applied to music source separation in Section 5.2, and the causal "Seq-U-Net" variant which is applied to generation tasks such as audio and music score generation in Section 5.3. By processing the raw audio waveform directly, Wave-U-Net takes a novel approach in the context of audio source separation and demonstrates it is competitive with spectrogram-based approaches. To mitigate the computational challenges for these models due to their high-dimensional input, we exploit the fact that many features of interest only change slowly relative to the input's sampling frequency by reducing the resolution of the convolutional layers. As a result, our models require less memory and computation time than comparable state-of-the-art approaches, such as Wavenet and TCN. In contrast to such models, which apply the same operation at each time-step during inference (time-invariant systems), Seq-U-Net performs different computations at different time-steps due to its down- and upsampling. Given the success of the Seq-U-Net, further exploration of time-varying systems might yield additional improvements in the future.

In summary, we presented a set of approaches that allows machine listening models to more flexibly process different types of data to avoid overfitting to potentially small datasets that are specifically annotated for the task at hand. These techniques are especially useful since the performance of DL models scales well with increasing training set sizes. To combat long training times and prohibitive memory usage that are exacerbated by large-scale training, we further developed architectural improvements to end-to-end models. Some of the proposed techniques are applicable beyond machine listening – most notably, model training in the face of certain missing data scenarios as shown in Chapter 3.

6.2 Future work

In this thesis, we focused on leveraging datasets of similar tasks and integrating incomplete datasets as a means of improving model generalisation. Using unlabelled data is another promising avenue for imbuing machine listening models with additional prior knowledge, especially since unlabelled audio data is often available in much larger quantities than audio data labelled for the task at hand. Inspiration could come from related areas that recently achieved immense successes using various pre-training approaches, such as natural language processing [Devlin et al., 2019] and computer vision [Chen et al., 2020]. In particular, auto-regressive sequence modelling as discussed in Section 2.6 appears to be a powerful pre-training objective – many relevant features are discovered by the model in an effort to predict future elements of a sequence given previous ones. This principle could be applied to the audio domain as well due to its time-dependent nature.

The adversarial learning framework we developed in Section 3.3 uses the standard GAN framework by Goodfellow et al. [2014], but employs additional regularisation on the discriminators in the form of spectral normalisation [Miyato et al., 2018] to stabilise training. Recently, more stable GAN variants such as the Wasserstein GAN [Arjovsky et al., 2017] were proposed that do not involve estimating probability density ratios and so do not allow manipulating probabilities in the same way we required to establish our framework. Therefore, it would be desirable to extend the framework to other GAN formulations as well. Factorising discriminators enables incorporating more prior knowledge into the design of neural architectures in GANs, which could improve empirical results in applied domains. The presented factorisation is generally applicable independent of model choice, so it can be readily integrated into many existing GAN-based approaches. Since the joint density can be factorised in different ways, multiple extensions are conceivable depending on the particular application (as shown in Section 3.3.3).

To expand on the studies from Section 4.3, meta learning could be used more specifically for few-shot learning, i.e., generalising from extremely few examples in only a few SGD update steps. Datasets could be extended to include a protocol defining a set of sub-tasks to establish a task distribution to use during meta learning, as well as a set of separate test tasks to enable comparison between approaches. For example, genre detection could be phrased as a few-shot problem where genres have to be recognised after training on a dataset with only very few music pieces per genre label. Overall, this could lead to more flexible machine listening models that do not rely on a fixed set of classes or circumstances, but can adapt quickly to new information.

To compare models capable of performing a wide variety of MIR tasks, a standardised multi-task benchmark for MIR should be established, similarly to the SuperGLUE benchmark found in natural language processing [Wang et al., 2019]. This would not only enable more accurate comparisons between such models but likely also spur more research in this area since it is more straightforward to compare new approaches with the state-of-the-art. Successively adding more MIR tasks and datasets into such a task collection would be very important for stable training and rigorous evaluation of meta- and multi-task learning approaches. With a larger set of training tasks, such approaches are much less likely to overfit to a particular set of tasks and more likely to generalise to new, unseen tasks – just as model generalisation improves when adding more data points to the training data, "meta-generalisation" improves with more training tasks, since a training task represents an individual data point in a "task dataset". Additionally, with more test tasks, the final performance on an unseen task can be estimated with higher precision, enabling a more rigorous evaluation.

Lastly, the sequence models proposed in Chapter 5 can be improved in various ways and analysed further. A limitation of our approach is that the levels of resolution along with the processing capacity at each resolution has to be manually pre-defined, which could limit performance. Future work could include potential solutions as used in the Phased LSTM [Neil et al., 2016] so the model can adapt its levels of resolution more dynamically to the task. To understand the inner workings of our models in more detail, one could investigate to which extent our models perform a spectral analysis for a given audio waveform, and how to incorporate computations similar to those in a multi-scale filterbank. For separation in the case of Wave-U-Net, one could also explicitly compute a decomposition of the input signal into a hierarchical set of basis signals and weightings on which to perform the separation, similar to the TasNet [Luo and Mesgarani, 2017]. Future work could also focus on improving the architecture of the down- and upsampling blocks. Attention mechanisms have shown great potential for sequence modelling and could be integrated into our approach by using attention operations in each down- and upsampling block alongside or instead of convolutions to further improve performance, as suggested by Child et al. [2019].

Seq-U-Net performs competitively on real-world benchmarks, while results on the synthetic copy task show that it can not retain high-frequency information over large numbers of time steps. This suggests only modelling long-term dependencies between "slow features" might be sufficient to achieve good performance in many application scenarios – although this should be investigated further in the future.

With the proliferation of end-to-end architectures for machine listening such as Wavenet and Wave-U-Net, research could also investigate suitable loss functions for such models that directly generate raw audio. For prediction tasks, we hypothesise that simple losses such as L1 and L2 losses reflect human perception relatively well when used to compare model predictions to targets in the spectrogram domain, but not when applying the same losses directly in the time domain. For example, if the model outputs an audio signal that is exactly identical to the target signal but has opposite phase, the resulting error will be large despite a negligible perceptual difference. Generative adversarial networks [Goodfellow et al., 2014, Stoller et al., 2018b] could be investigated in this context, since their error is dependent on a separate discriminator network that aims to distinguish model output and target. They also represent a possible approach for generating audio directly in the time domain. Alternatively, spectrogram-based losses could be integrated into models with raw waveform outputs by computing spectrogram representations of model output and target waveform, computing the desired loss in the spectrogram domain, and then backpropagating the gradient of the loss through the spectrogram computation operation.

In this thesis, we developed methods to integrate additionally available data of various kinds into the training process of DNNs to increase their generalisation capability, and demonstrated performance improvements in various MIR tasks such as audio source separation and singing voice detection. These advances will hopefully lay the groundwork for a new era of MIR, in which DNNs are efficiently regularised through data-driven priors similar to the ones we proposed, to obtain good generalisation for a large number of different tasks despite having only few labelled training data. Instead of training models from scratch (i.e. a random initialisation of parameters), we hope our initial studies on pre-training models (in our case, through meta-learning) will spur further research in this direction so that starting with a pre-trained model and then fine-tuning it to the task of interest will be the new default. This will not only improve generalisation performance, but is also more time and energy efficient due to the low amount of task-specific training required. Most current research in DL for audio and music signals spends immense efforts on designing task-specific DNN architectures. In this context, we believe that our Wave-U-Net and Seq-U-Net models proposed in Chapter 5 present a substantial step towards more general DNN architectures. For example, the Wave-U-Net was also trained to successfully perform speech enchancement [Macartney and Weyde, 2018] and slightly adapted to perform lyrics alignment [Stoller et al., 2019], while the Seq-U-Net was tested across several different tasks in Section 5.3. Obtaining such more generally applicable DNN architectures not only saves time otherwise needed for architecture design, but also provides us with models that could be trained to perform a variety of

different tasks with only little adaptation of the network architecture parameters, using meta-learning, self-supervised learning and other approaches discussed in this thesis.

Appendix A

Loss function derivation for joint SVS and SVD

In this appendix, we will theoretically derive a probabilistically grounded loss function for the problem of joint singing voice detection and separation.

Let m be the input spectrogram to the model represented as a vector of dimensionality $T \cdot F$, where T is the number of time frames, and F is the number of frequency bins in the spectrogram. Let s describe the combination of an accompaniment and voice source spectrogram excerpt with a vector of dimensionality $d = 2 \cdot T \cdot F$. Furthermore, let $o \in \{0, 1\}^T$ be a binary vector of dimensionality T describing vocal activity labels at each time frame of an audio excerpt m.

We define a model with parameters ϕ that yields the probability distribution $p_\phi(s, o|m)$ mapping mixtures to accompaniment and vocal tracks and predict vocal activity. Following the Multi-Task learning principle, the model internally calculates a hidden representation so that the outputs become independent of each other under this hidden representation: $p_\phi(s, o|m) = p_\phi(s|h) \cdot p_\phi(o|h) \cdot p_\phi(h|m)$. We will focus on a deterministic prediction of h using a function $f_\phi^h : m \to h$.

We now consider the overall joint probability $p_\phi(m, h, s, o)$, viewing our model as a data generator that we can train according to Maximum Likelihood. Since we always condition on the mixture input m, we set $p(m)$ to the true distribution of mixtures so that our model always predicts based on real mixture inputs which are not modeled itself. Therefore, we can decompose the joint probability as $p_\phi(m, h, s, o) = p_D(m) \cdot p_\phi(s, o|m)$, where p_D is the empirical data distribution, and our parameters only model the conditional likelihood.

Now assume we have a multi-track dataset and mixtures with vocal activity labels available with N and M observations, respectively. Applying the principle

of Maximum likelihood for a generative model with missing data, we can define the following log-likelihood function:

$$L_\phi = \sum_{i=1}^{N} \log p_\phi(s_i, m_i) + \sum_{j=1}^{M} \log p_\phi(o_j, m_j) \tag{A.1}$$

We compute the above marginal probabilities by marginalising out the respectively missing variables. For the first term, this means

$$\begin{aligned}
\log p_\phi(s, m) &= \log \int_h \sum_o p(m) p_\phi(s, o|m) dh \\
&= \log p_D(m) + \log \int_h \sum_o [p_\phi(s|h) \cdot p_\phi(o|h) \cdot p_\phi(h|m)] dh \\
&\propto_\phi \log \int_h [p_\phi(s|h) p_\phi(h|m)] dh \\
&\approx \log p_\phi(s|f_\phi^h(m)) \\
&:= \log \mathcal{N}(s|f_\phi^s(f_\phi^h(m_i)), \sigma^2 \mathbf{I}_d)
\end{aligned} \tag{A.2}$$

Integration and summation over h and o respectively occurs over all its dimensions. Note that the dependency on o disappears, since we integrate $p(o|h)$ over all o. In the second last line, the sampling over h is replaced by the output of the deterministic function f_ϕ^h. In the last line, we further define the output likelihood over the joint sources as a normal distribution with a mean given as output of our model, and a learnable scalar σ^2 as isotropic variance in each dimension. $\mathbf{I}_d$ denotes the $d \times d$ identity matrix.

For the likelihood of mixtures with vocal activity labels $p_\phi(o, m)$, we can make similar simplifications. Firstly, we set the probability of a vocal activity curve to be equal to the product of probabilities at each time step t:

$$\begin{aligned}
p_\phi(o, m) &= \prod_{t=1}^{T} p_\phi(o^t, m) \\
\Rightarrow \log p_\phi(o, m) &= \sum_{t=1}^{T} \log p_\phi(o^t, m)
\end{aligned} \tag{A.3}$$

Then, we compute $\log p_\phi(o^t, m)$ as follows:

$$
\begin{aligned}
\log p_\phi(o^t, m) &\propto_\phi \log \int_h [p_\phi(o^t|h) p_\phi(h|m)] dh \\
&\approx \log p_\phi(o^t | f_\phi^h(m)) \\
&= o^t \cdot \log \sigma(f_\phi^o(f_\phi^h(m))) + (1 - o^t) \cdot \log(1 - \sigma(f_\phi^o(f_\phi^h(m)))) \\
&= -H(o^t, \sigma(f_\phi^o(f_\phi^h(m))))
\end{aligned}
\tag{A.4}
$$

The binary cross-entropy is denoted by H. When defining $p_\phi(o^t|m)$ to be $\sigma(f_\phi^o(f_\phi^h(m)))$ with σ as sigmoid function, we arrive at minimising the binary cross-entropy between label o^t and prediction at each time step t.

We combine the above terms for separation predictions with a Normal likelihood from (A.2) and the binary cross-entropy terms from (A.4) to determine the log-likelihood from (A.1) on the combined data:

$$
\begin{aligned}
L_\phi &= \sum_{i=1}^{N} \log p_\phi(s_i, m_i) + \sum_{j=1}^{M} \sum_{t=1}^{T} \log p_\phi(o_j, m_j) \\
&\propto_\phi \sum_{i=1}^{N} \log \mathcal{N}(s | f_\phi^s(f_\phi^h(m_i)), \sigma^2 \mathbf{I}_d) - \sum_{j=1}^{M} \sum_{t=1}^{T} H(o_j^t, \sigma(f_\phi^o(f_\phi^h(m_j)))) \\
&\propto_\phi \sum_{i=1}^{N} (-\frac{1}{2\sigma^2} ||s_i - f_\phi^s(m_i)||^2 - d \log \sigma) - \sum_{j=1}^{M} \sum_{t=1}^{T} H(o_j^t, \sigma(f_\phi^o(f_\phi^h(m_j)))) \\
&= -\frac{1}{2\sigma^2} \sum_{i=1}^{N} ||s_i - f_\phi^s(m_i)||^2 - Nd \log \sigma - \sum_{j=1}^{M} \sum_{t=1}^{T} H(o_j^t, \sigma(f_\phi^o(f_\phi^h(m_j))))
\end{aligned}
\tag{A.5}
$$

Note that we include σ^2 as a learnable parameter so we cannot further simplify from here.

One issue with directly using this loss function for training is the scaling. With more data points (larger N and M), or with a larger dimensionality d of the source output space, the loss varies more, since we have more summands in the former case and more summation over quadratic differences in the latter case. When using the loss for stochastic gradient descent, this means that the size of gradients varies strongly depending on the usage scenario, which can impede learning. To make the loss independent of dataset size and dimensionality of the source output, we divide by $N + M$ and d:

$$
\begin{aligned}
&\hat{L}_\phi := \frac{L_\phi}{(N+M)d} = \\
&-\frac{1}{2\sigma^2(N+M)} \sum_{i=1}^{N} MSE(s_i, f_\phi^s(f_\phi^h(m_i))) \\
&-\frac{N}{N+M} \log\sigma \\
&-\frac{1}{2F}\frac{1}{N+M} \sum_{j=1}^{M} \frac{1}{T} \sum_{t=1}^{T} H(o_j^t, \sigma(f_\phi^o(f_\phi^h(m_j))))
\end{aligned}
\tag{A.6}
$$

MSE defines the mean squared error between source ground truth and prediction. Due to division by $d = 2TF$, the cross-entropy term for each sample m_i can now be seen as the average over the cross-entropy terms for each vocal activity label o^t.

We can write the naive loss combining MSE and average cross-entropy, as used in the main paper, as

$$
I = \alpha\frac{1}{N} \sum_{i=1}^{N} MSE(s_i, f_\phi^s(f_\phi^h(m_i))) + (1-\alpha)\frac{1}{M} \sum_{j=1}^{M} \frac{1}{T} \sum_{t=1}^{T} H(o_j^t, \sigma(f_\phi^o(f_\phi^h(m_j))))
\tag{A.7}
$$

We observe that maximising our scaled log-likelihood from (A.6) is equivalent to minimising the naive loss, when fixing the variance parameter σ^2 and the weighting α to a specific value:

$$
\begin{aligned}
\hat{L}_\phi &= -I - \frac{N}{N+M}\log\sigma \text{ if} \\
\alpha &= \frac{1}{2\sigma^2}\frac{N}{N+M} \text{ and} \\
1-\alpha &= \frac{1}{2F}\frac{M}{N+M} \text{ so that} \\
\sigma^2 &= \frac{FN}{2FN+2MF-M}
\end{aligned}
\tag{A.8}
$$

Although our log-likelihood function is theoretically more satisfying as it avoids a hyper-parameter and naturally accounts for a different amount of samples in both datasets, empirically we did not achieve better performance using this loss. A reason might be the learning dynamics of an ADAM optimiser on a neural network combined with the loss scaling: As the learnable variance σ decreases during training, the loss grows in size and variance far beyond that of the naive loss, which might mean lower learning rates are needed. This should be investigated further.

Bibliography

Anna Aljanaki, Frans Wiering, and Remco C Veltkamp. Studying emotion induced by music through a crowdsourcing game. *Information Processing & Management*, 52(1):115–128, 2016.

Moray Allan and Christopher Williams. Harmonising Chorales by Probabilistic Inference. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 25–32. MIT Press, 2005.

Amjad Almahairi, Sai Rajeswar, Alessandro Sordoni, Philip Bachman, and Aaron Courville. Augmented CycleGAN: Learning Many-to-Many Mappings from Unpaired Data. *CoRR*, abs/1802.10151, 2018.

Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2017.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *CoRR*, abs/1701.07875, 2017.

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Yoshua Bengio, et al. End-to-end attention-based large vocabulary speech recognition. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4945–4949. IEEE, 2016.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Convolutional Sequence Modeling Revisited. In *International Conference on Learning Representations (ICLR) Workshop Track*, 2018.

Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: Challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2): 157–166, 1994. ISSN 1045-9227. DOI: 10.1109/72.279181.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances In Neural Information Processing Systems*, pages 153–160, 2007.

Yoshua Bengio, Aaron Courville, and Pierre Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828, 2013.

Rachel Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Bello. MedleyDB: A multitrack dataset for annotation-intensive MIR research. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 155–160, 2014.

Rachel M Bittner, Brian McFee, Justin Salamon, Peter Li, and Juan Pablo Bello. Deep Salience Representations for F0 Estimation in Polyphonic Music. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 63–70, 2017.

Rachel M. Bittner, Brian McFee, and Juan P. Bello. Multitask Learning for Fundamental Frequency Estimation in Music. *CoRR*, abs/1809.00381, September 2018.

Dawn A. A. Black, Ma Li, and Mi Tian. Automatic Identification of Emotional Cues in Chinese Opera Singing. In *13th Int. Conf. on Music Perception and Cognition (ICMPC)*, pages 250–255, 2014.

Sebastian Böck, Florian Krebs, and Gerhard Widmer. Joint beat and downbeat tracking with recurrent neural networks. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 255–261, 2016.

Dmitry Bogdanov, Minz Won, Philip Tovstogan, Alastair Porter, and Xavier Serra. The MTG-Jamendo dataset for automatic music tagging. In *Machine Learning for Music Discovery Workshop at the International Conference on Machine Learning (ICML 2019)*, Long Beach, CA, United States, June 2019.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *Proc. of the International Conference on Machine Learning (ICML)*, pages 1159–1166, 2012.

Philemon Brakel and Yoshua Bengio. Learning Independent Features with Adversarial Nets for Non-linear ICA. *CoRR*, abs/1710.05050, 2017.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis. *CoRR*, abs/1809.11096, 2019.

Victor Campos, Brendan Jou, Xavier Giro-i-Nieto, Jordi Torres, and Shih-Fu Chang. Skip RNN: Learning to skip state updates in recurrent neural networks. *CoRR*, abs/1708.06834, 2017.

Rich Caruana. Multitask Learning. In Sebastian Thrun and Lorien Pratt, editors, *Learning to Learn*, pages 95–133. Springer US, Boston, MA, 1998. ISBN 978-1-4615-5529-2.

A Taylan Cemgil, Cédric Févotte, and Simon J Godsill. Variational and stochastic inference for Bayesian source separation. *Digital Signal Processing*, 17(5):891–913, 2007.

Tak-Shing Chan, Tzu-Chun Yeh, Zhe-Cheng Fan, Hung-Wei Chen, Li Su, Yi-Hsuan Yang, and Roger Jang. Vocal activity informed singing voice separation with the iKala dataset. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 718–722. IEEE, 2015.

Shiyu Chang, Yang Zhang, Wei Han, Mo Yu, Xiaoxiao Guo, Wei Tan, Xiaodong Cui, Michael Witbrock, Mark A Hasegawa-Johnson, and Thomas S Huang. Dilated Recurrent Neural Networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 77–87. Curran Associates, Inc., 2017.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. *CoRR*, abs/2002.05709, 2020.

Tsung-Ping Chen, Li Su, et al. Functional harmony recognition of symbolic music data with multi-task recurrent neural networks. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 90–97, 2018.

Bhusan Chettri, Daniel Stoller, Veronica Morfi, Marco A. Martínez Ramírez, Emmanouil Benetos, and Bob L. Sturm. Ensemble models for spoofing detection in automatic speaker verification. In *Proceedings of INTERSPEECH*, pages 1018–1022, 2019.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating Long Sequences with Sparse Transformers. *CoRR*, abs/1904.10509, 2019.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Repre-

sentations using RNN Encoder-Decoder for Statistical Machine Translation. *abs/1406.1078*, CoRR, 2014.

Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Transfer learning for music classification and regression tasks. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, volume 18, pages 141–149, 2018.

Jan Chorowski, Ron J. Weiss, Samy Bengio, and Aäron van den Oord. Unsupervised speech representation learning using WaveNet autoencoders. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(12):2041–2053, 2019. ISSN 2329-9290, 2329-9304. DOI: 10.1109/TASLP.2019.2938863.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *abs/1412.3555*, CoRR, December 2014.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical Multiscale Recurrent Neural Networks. *CoRR*, abs/1609.01704, 2016.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Débora C Corrêa and Francisco Ap Rodrigues. A survey on symbolic data-based music genre classification. *Expert Systems with Applications*, 60:190–210, 2016.

Jason Cramer, Ho-Hsiang Wu, Justin Salamon, and Juan Pablo Bello. Look, listen, and learn more: Design choices for deep audio embeddings. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3852–3856. IEEE, 2019.

Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. Generative Adversarial Networks: An Overview. *IEEE Signal Processing Magazine*, 35(1):53–65, January 2018. ISSN 1053-5888. DOI: 10.1109/MSP.2017.2765202.

Elizabeth R DeLong, David M DeLong, and Daniel L Clarke-Pearson. Comparing the areas under two or more correlated receiver operating characteristic curves: A nonparametric approach. *Biometrics*, pages 837–845, 1988.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR09*, 2009.

Li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8599–8603. IEEE, 2013.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, abs/1810.04805, 2019.

Sander Dieleman, Aäron van den Oord, and Karen Simonyan. The challenge of realistic music generation: Modelling raw audio at scale. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 8000–8010, 2018.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

Clement S. J. Doire and Olumide Okubadejo. Interleaved Multitask Learning for Audio Source Separation with Independent Databases. *CoRR*, abs/1908.05182, 2019.

Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

Salah El Hihi and Yoshua Bengio. Hierarchical Recurrent Neural Networks for Long-term Dependencies. In *Proceedings of the 8th International Conference on Neural Information Processing Systems*, NIPS'95, pages 493–499. MIT Press, 1995.

Sebastian Ewert and Mark B Sandler. Structured Dropout for Weak Label and Multi-Instance Learning and Its Application to Score-Informed Source Separation. *arXiv preprint arXiv:1609.04557*, 2016.

Sebastian Ewert, Bryan Pardo, Meinard Müller, and Mark D. Plumbley. Score-Informed Source Separation for Musical Audio Recordings: An Overview. *IEEE Signal Processing Magazine*, 31(3):116–124, 2014. ISSN 1053-5888. DOI: 10.1109/MSP.2013.2296076.

Cédric Févotte. Bayesian audio source separation. In *Blind Speech Separation*, pages 305–335. Springer, 2007.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In Doina Precup and Yee Whye Teh, editors, *Proc. of the International Conference on Machine Learning (ICML)*,

volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135, International Convention Centre, Sydney, Australia, August 2017. PMLR.

J. T. Foote and M. L. Cooper. Media segmentation using self-similarity decomposition. In *Electronic Imaging 2003*, pages 167–175. International Society for Optics and Photonics, 2003.

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, April 1980. ISSN 0340-1200, 1432-0770. DOI: 10.1007/BF00344251.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proc. of the International Conference on Machine Learning (ICML)*, volume 48, pages 1050–1059, 2016.

Zhe Gan, Liqun Chen, Weiyao Wang, Yunchen Pu, Yizhe Zhang, Hao Liu, Chunyuan Li, and Lawrence Carin. Triangle Generative Adversarial Networks. *CoRR*, abs/1709.06548, 2017.

Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio Set: An ontology and human-labeled dataset for audio events. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, LA, 2017.

Olivier Gillet and Gaël Richard. ENST-Drums: An extensive audio-visual database for drum signals processing. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 156–159, 2006.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 580–587, USA, 2014. IEEE Computer Society. ISBN 978-1-4799-5118-5. DOI: 10.1109/CVPR.2014.81.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. RWC music database: Popular, classical and jazz music databases. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 287–288, 2002.

Emad M Grais, Dominic Ward, and Mark D Plumbley. Raw Multi-Channel Audio Source Separation using Multi-Resolution Convolutional Auto-Encoders. *arXiv preprint arXiv:1803.00702*, 2018.

Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6645–6649, 2013.

D. Griffin and Jae Lim. Signal estimation from modified short-time Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984. ISSN 0096-3518. DOI: 10.1109/TASSP.1984.1164317.

Thomas Grill and Jan Schlüter. Music boundary detection using neural networks on combined features and two-level annotations. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 531–537, 2015.

Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved Training of Wasserstein GANs. *CoRR*, abs/1704.00028, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking ImageNet Pre-training. *CoRR*, abs/1811.08883, 2018.

Toni Heittola, Annamaria Mesaros, Tuomas Virtanen, and Antti Eronen. Sound event detection in multisource environments using source separation. In *Machine Listening in Multisource Environments*, 2011.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances In Neural Information Processing Systems*, pages 6629–6640, 2017.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

Po-Sen Huang, Scott Deeann Chen, Paris Smaragdis, and Mark Hasegawa-Johnson. Singing-voice separation from monaural recordings using robust principal component analysis. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 57–60. IEEE, 2012.

Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis. Singing-Voice Separation from Monaural Recordings using Deep Recurrent Neural Networks. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 477–482, 2014.

Ferenc Huszár. How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary? *CoRR*, abs/1511.05101, 2015.

Yukara Ikemiya, Kazuyoshi Yoshii, and Katsutoshi Itoyama. Singing voice analysis and editing based on mutually dependent F0 estimation and source separation. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 574–578. IEEE, 2015.

Daniel Jiwoong Im, Alllan He Ma, Graham W. Taylor, and Kristin Branson. Quantitatively Evaluating GANs With Divergences Proposed for Training. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2018.

IMIRSEL. Music Information Retrieval Exchange (MIREX). https://www.music-ir.org/mirex/wiki/MIREX_HOME, March 2020.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. *CoRR*, abs/1611.07004, 2016.

Jan Jakubik and Halina Kwaśnicka. Similarity-Based Summarization of Music Files for Support Vector Machines. *Complexity*, 2018:1935938, 2018. ISSN 1076-2787. DOI: 10.1155/2018/1935938.

Andreas Jansson, Eric J. Humphrey, Nicola Montecchio, Rachel Bittner, Aparna Kumar, and Tillman Weyde. Singing Voice Separation with Deep U-Net Convolutional Networks. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 323–332, 2017.

Zeyu Jin, Adam Finkelstein, Gautham J. Mysore, and Jingwan Lu. FFTNet: A Real-Time Speaker-Dependent Neural Vocoder. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural Machine Translation in Linear Time. *CoRR*, abs/1610.10099, 2016.

Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron van den Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient Neural Audio Synthesis. In Jennifer Dy and Andreas Krause, editors, *Proc. of the International Conference on Machine Learning (ICML)*, volume 80, pages 2410–2419. PMLR, 2018.

Theofanis Karaletsos. Adversarial Message Passing For Graphical Models. *CoRR*, abs/1612.05048, 2016.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2018a.

Tero Karras, Samuli Laine, and Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. *CoRR*, abs/1812.04948, 2018b.

Rainer Kelz, Sebastian Böck, and Cierhard Widnaer. Multitask learning for polyphonic piano transcription, a case study. In *International Workshop on Multilayer Music Representation and Processing (MMRP)*, pages 85–91. IEEE, 2019.

Jaehun Kim, Julián Urbano, Cynthia CS Liem, and Alan Hanjalic. One deep music representation to rule them all? A comparative analysis of different representation learning strategies. *Neural Computing and Applications*, pages 1–27, 2019.

Suyoun Kim, Takaaki Hori, and Shinji Watanabe. Joint CTC-Attention based End-to-End Speech Recognition using Multi-task Learning. *CoRR*, abs/1609.06773, 2016.

Peter Knees, Ángel Faraldo, Perfecto Herrera, Richard Vogl, Sebastian Böck, Florian Hörschläger, and Mickael Le Goff. Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR'15)*, Málaga, Spain, 2015.

Qiuqiang Kong, Yong Xu, Wenwu Wang, and Mark D. Plumbley. A joint separation-classification model for sound event detection of weakly labelled data. *CoRR*, abs/1711.03037, 2017.

Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D. Plumbley. PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition. *CoRR*, abs/1912.10211, 2020.

Jan Koutník, Klaus Greff, Faustino Gomez, and Jürgen Schmidhuber. A Clockwork RNN. *CoRR*, abs/1402.3511, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

Jonathan Le Roux, Nobutaka Ono, and Shigeki Sagayama. Explicit consistency constraints for STFT spectrograms and their application to phase reconstruction. In *SAPA@ INTERSPEECH*, pages 23–28, 2008.

Jongpil Lee, Jiyoung Park, Keunhyoung Luke Kim, and Juhan Nam. Sample-level Deep Convolutional Neural Networks for Music Auto-tagging Using Raw Waveforms. *CoRR*, abs/1703.01789, 2017.

Kyungyun Lee, Keunwoo Choi, and Juhan Nam. Revisiting Singing Voice Detection: A quantitative review and the future outlook. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 506–513, 2018.

Zhen-Hua Ling, Shi-Yin Kang, Heiga Zen, Andrew Senior, Mike Schuster, Xiao-Jun Qian, Helen M Meng, and Li Deng. Deep learning for acoustic modeling in parametric speech generation: A systematic review of existing techniques and future trends. *IEEE Signal Processing Magazine*, 32(3):35–52, 2015.

Seppo Linnainmaa. Taylor expansion of the accumulated rounding error. *BIT*, 16(2):146–160, 1976. ISSN 0006-3835, 1572-9125. DOI: 10.1007/BF01931367.

Antoine Liutkus, Derry Fitzgerald, and Zafar Rafii. Scalable audio separation with light kernel additive modelling. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 76–80. IEEE, 2015.

Antoine Liutkus, Fabian-Robert Stöter, Zafar Rafii, Daichi Kitamura, Bertrand Rivet, Nobutaka Ito, Nobutaka Ono, and Julie Fontecave. The 2016 signal separation evaluation campaign. In *Proceedings of the International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA)*, pages 323–332, 2017.

Y. Luo, Z. Chen, J. R. Hershey, J. Le Roux, and N. Mesgarani. Deep clustering and conventional networks for music separation: Stronger together. In *Proc. of*

the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 61–65, 2017. DOI: 10.1109/ICASSP.2017.7952118.

Yi Luo and Nima Mesgarani. TasNet: Time-domain audio separation network for real-time, single-channel speech separation. *CoRR*, abs/1711.00541, 2017.

Craig Macartney and Tillman Weyde. Improved Speech Enhancement with the Wave-U-Net. *CoRR*, abs/1811.11307, 2018.

David JC MacKay. Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 354(1):73–80, 1995.

Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, and Stephen Paul Smolley. Least Squares Generative Adversarial Networks. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

Ugo Marchand, Quentin Fresnel, and Geoffroy Peeters. GTZAN-Rhythm: Extending the GTZAN test-set with beat, downbeat and swing annotations. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2015.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a Large Annotated Corpus of English: The Penn Treebank. *Comput. Linguist.*, 19(2):313–330, 1993. ISSN 0891-2017.

Matthias Mauch and Simon Dixon. pYIN: A fundamental frequency estimator using probabilistic threshold distributions. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 659–663. IEEE, 2014.

Matthew C. McCallum. Unsupervised Learning of Deep Features for Music Segmentation. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 346–350, Brighton, United Kingdom, May 2019. IEEE. ISBN 978-1-4799-8131-1. DOI: 10.1109/ICASSP.2019.8683407.

Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. SampleRNN: An Unconditional End-to-End Neural Audio Generation Model. *CoRR*, abs/1612.07837, 2016.

Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. TUT database for acoustic scene classification and sound event detection. In *2016 24th European Signal Processing Conference (EUSIPCO)*, pages 1128–1132. IEEE, 2016.

Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for GANs do actually converge? In Jennifer Dy and Andreas Krause, editors, *Proc. of the International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 3481–3490, Stockholmsmässan, Stockholm Sweden, 2018. PMLR.

Luke Metz, Niru Maheswaranathan, Brian Cheung, and Jascha Sohl-Dickstein. Learning unsupervised learning rules. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019.

Marius Miron, Jordi Janer Mestres, and Emilia Gómez Gutiérrez. Generating data to train convolutional neural networks for classical music source separation. In *Proceedings of the 14th Sound and Music Computing Conference*. Aalto University, 2017.

Saumitra Mishra, Daniel Stoller, Emmanouil Benetos, Bob L. Sturm, and Simon Dixon. GAN-based generation and automatic selection of explanations for neural networks. In *Safe Machine Learning Workshop at the International Conference on Learning Representations (ICLR)*, 2019.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional smoothing with virtual adversarial training. *CoRR*, abs/1507.00677, 2015.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks. *CoRR*, abs/1802.05957, 2018.

Olof Mogren. C-RNN-GAN: A continuous recurrent neural network with adversarial training. In *Constructive Machine Learning Workshop (CML) at NIPS*, 2016.

Noam Mor, Lior Wolf, Adam Polyak, and Yaniv Taigman. A Universal Music Translation Network. *CoRR*, abs/1805.07848, 2018.

Gnostothea-Veroniki Morfi. *Automatic Detection and Classification of Bird Sounds in Low-Resource Wildlife Audio Datasets*. PhD thesis, Quen Mary University of London, 2019.

Jared S. Murray. Multiple Imputation: A Review of Practical and Theoretical Findings. *CoRR*, abs/1801.04058, 2018.

Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased LSTM: Accelerating Recurrent Network Training for Long or Event-based Sequences. *CoRR*, abs/1610.09513, 2016.

Alex Nichol, Joshua Achiam, and John Schulman. On First-Order Meta-Learning Algorithms. *CoRR*, abs/1803.02999, 2018.

I. Nolasco, A. Terenzi, S. Cecchi, S. Orcioni, H. L. Bear, and E. Benetos. Audio-based identification of beehive states. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8256–8260, 2019.

Aditya Arie Nugraha, Antoine Liutkus, and Emmanuel Vincent. *Multichannel Audio Source Separation with Deep Neural Networks*. PhD Thesis, Inria, 2015.

Augustus Odena. Semi-Supervised Learning with Generative Adversarial Networks. *CoRR*, abs/1606.01583, 2016.

Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and Checkerboard Artifacts. *Distill*, 2016. DOI: 10.23915/distill.00003.

Alexey Ozerov, Emmanuel Vincent, and Frédéric Bimbot. A general flexible framework for the handling of prior information in audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(4):1118–1133, 2012.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

Santiago Pascual, Antonio Bonafonte, and Joan Serra. SEGAN: Speech enhancement generative adversarial network. *CoRR*, abs/1703.09452, 2017.

Santiago Pascual, Mirco Ravanelli, Joan Serrà, Antonio Bonafonte, and Yoshua Bengio. Learning Problem-agnostic Speech Representations from Multiple Self-supervised Tasks. *CoRR*, abs/1904.03416, 2019.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, volume 1, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. DOI: 10.18653/v1/N18-1202.

Yunchen Pu, Shuyang Dai, Zhe Gan, Weiyao Wang, Guoyin Wang, Yizhe Zhang, Ricardo Henao, and Lawrence Carin Duke. JointGAN: Multi-Domain Joint Distribution Learning with Generative Adversarial Nets. In Jennifer Dy and Andreas Krause, editors, *Proc. of the International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 4151–4160, Stockholmsmässan, Stockholm Sweden, July 2018. PMLR.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *CoRR*, abs/1511.06434, 2015.

Colin Raffel, Brian Mcfee, Eric Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel Ellis. Mir_eval: A transparent implementation of common MIR metrics. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

Zafar Rafii, François Germain, Dennis L Sun, and Gautham J Mysore. Combining Modeling Of Singing Voice And Background Music For Automatic Separation Of Musical Mixtures. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, volume 10, pages 645–680, 2013.

Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. The Musdb18 Corpus For Music Separation, 2017.

Mathieu Ramona, Gaël Richard, and Bertrand David. Vocal detection in music with Support Vector Machines. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1885–1888, 2008.

Dario Rethage, Jordi Pons, and Xavier Serra. A Wavenet for Speech Denoising. *CoRR*, abs/1706.07162, 2017.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-Agnostic Interpretability of Machine Learning. *CoRR*, abs/ preprint, 2016.

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proc. of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

Haşim Sak, Andrew Senior, Kanishka Rao, Ozan Irsoy, Alex Graves, Françoise Beaufays, and Johan Schalkwyk. Learning acoustic frame labeling for speech recognition with recurrent neural networks. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4280–4284, 2015.

J. Schlüter. Learning to pinpoint singing voice from weakly labeled examples. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 44–50, 2016.

J. Schlüter and S. Böck. Improved musical onset detection with Convolutional Neural Networks. In *Proc. of the IEEE International Conference on Acoustics,*

Speech and Signal Processing (ICASSP), pages 6979–6983, May 2014. DOI: 10.1109/ICASSP.2014.6854953.

Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. Wav2vec: Unsupervised Pre-training for Speech Recognition. *CoRR*, abs/1904.05862, April 2019.

Hendrik Schreiber. A single-step approach to musical tempo estimation using a convolutional neural network. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 95–105, 2018.

Evan Shelhamer, Kate Rakelly, Judy Hoffman, and Trevor Darrell. Clockwork Convnets for Video Semantic Segmentation. *CoRR*, abs/1608.03609, 2016.

Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. An End-to-End Neural Network for Polyphonic Music Transcription. *CoRR*, abs/1508.01774:555–560, 2015.

Andrew JR Simpson, Gerard Roma, and Mark D Plumbley. Deep karaoke: Extracting vocals from musical mixtures using a convolutional deep neural network. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 429–436. Springer, 2015.

Jeffrey C. Smith. *Correlation Analyses of Encoded Music Performance*. PhD Thesis, Stanford University, Stanford, CA, USA, 2013.

Jordan Bennett Louis Smith, John Ashley Burgoyne, Ichiro Fujinaga, David De Roure, and J Stephen Downie. Design and creation of a large-scale database of structural annotations. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2011.

Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised map inference for image super-resolution. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2017.

Jost Tobias Springenberg. Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks. *CoRR*, abs/1511.06390, 2015.

D. Stoller, V. Akkermans, and S. Dixon. Detection of Cut-Points for Automatic Music Rearrangement. In *2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2018a. DOI: 10.1109/MLSP.2018.8516706.

Daniel Stoller and Simon Dixon. Analysis and classification of phonation modes in singing. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, volume 17, pages 80–86, 2016.

Daniel Stoller, Sebastian Ewert, and Simon Dixon. Adversarial Semi-Supervised Audio Source Separation applied to Singing Voice Extraction. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2391–2395, Calgary, Canada, 2018b. IEEE.

Daniel Stoller, Sebastian Ewert, and Simon Dixon. Jointly Detecting and Separating Singing Voice: A Multi-Task Approach. In Yannick Deville, Sharon Gannot, Russell Mason, Mark D. Plumbley, and Dominic Ward, editors, *Latent Variable Analysis and Signal Separation*, pages 329–339, 2018c.

Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-U-Net: A Multi-Scale Neural Network for End-to-End Source Separation. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, volume 19, pages 334–340, 2018d.

Daniel Stoller, Igor Vatolkin, and Heinrich Müller. Intuitive and efficient computer-aided music rearrangement with optimised processing of audio transitions. *Journal of New Music Research*, 47(5):416–437, 2018e. DOI: 10.1080/09298215.2018.1473448.

Daniel Stoller, Simon Durand, and Sebastian Ewert. End-to-end Lyrics Alignment for Polyphonic Music Using An Audio-to-Character Recognition Model. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK, 2019. IEEE.

Daniel Stoller, Sebastian Ewert, and Simon Dixon. Training Generative Adversarial Networks from Incomplete Observations using Factorised Discriminators. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020a.

Daniel Stoller, Mi Tian, Sebastian Ewert, and Simon Dixon. Seq-U-Net: A One-Dimensional Causal U-Net for Efficient Sequence Modelling. In *Proc. of the International Joint Conference on Artificial Intelligence - Pacific Rim International Conference on Artificial Intelligence (IJCAI-PRICAI)*, 2020b.

Fabian-Robert Stöter, Antoine Liutkus, and Nobutaka Ito. The 2018 Signal Separation Evaluation Campaign. *abs/1804.06267*, CoRR, July 2018.

D. Stowell, M. Wood, Y. Stylianou, and H. Glotin. Bird detection in audio: A survey and a challenge. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2016.

Bob L Sturm. A survey of evaluation in music genre recognition. In *International Workshop on Adaptive Multimedia Retrieval*, pages 29–66. Springer, 2012.

Dennis L Sun and Gautham J Mysore. Universal speech models for speaker independent single channel source separation. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 141–145. IEEE, 2013.

Niko Sünderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, et al. The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37(4-5):405–420, 2018.

Eleni Triantafillou, Hugo Larochelle, Jake Snell, Josh Tenenbaum, Kevin Jordan Swersky, Mengye Ren, Richard Zemel, and Sachin Ravi. Meta-learning for semi-supervised few-shot classification. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2018.

Trieu H. Trinh, Andrew M. Dai, Minh-Thang Luong, and Quoc V. Le. Learning Longer-term Dependencies in RNNs with Auxiliary Losses. *CoRR*, abs/1803.00144, 2018.

Trieu H. Trinh, Minh-Thang Luong, and Quoc V. Le. Selfie: Self-supervised Pretraining for Image Embedding. *CoRR*, abs/:1906.02940, 2019.

Soumya Tripathy, Juho Kannala, and Esa Rahtu. Learning image-to-image translation using paired and unpaired training samples. *CoRR*, abs/805.03189, 2018.

George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

S. Uhlich, M. Porcu, F. Giron, M. Enenkl, T. Kemp, N. Takahashi, and Y. Mitsufuji. Improving music source separation based on deep neural networks through data augmentation and network blending. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 261–265, March 2017. DOI: 10.1109/ICASSP.2017.7952158.

Stefan Uhlich, Franck Giron, and Yuki Mitsufuji. Deep neural network based instrument extraction from music. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2135–2139. IEEE, 2015.

Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Transfer learning by supervised pre-training for audio-based music classification. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. *CoRR*, abs/1609.03499, 2016.

Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C. Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis. Parallel WaveNet: Fast High-Fidelity Speech Synthesis. *CoRR*, abs/1711.10433, 2017.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding. *CoRR*, abs/1807.03748, 2019.

Sean Vasquez and Mike Lewis. MelNet: A Generative Model for Audio in the Frequency Domain. *CoRR*, abs/1906.01083, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *CoRR*, abs/1706.03762, 2017.

Igor Vatolkin and Daniel Stoller. Evolutionary multi-objective training set selection of data instances and augmentations for vocal detection. In *8th International Conference on Computational Intelligence in Music, Sound, Art and Design (EvoMUSART)*, pages 201–216, 2019. DOI: 10.1007/978-3-030-16667-0_14.

Shrikant Venkataramani and Paris Smaragdis. End-to-end Source Separation with Adaptive Front-Ends. *CoRR*, abs/1705.02514, 2017.

E. Vincent, R. Gribonval, and C. Fevotte. Performance measurement in blind audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(4):1462–1469, 2006. ISSN 1558-7916. DOI: 10.1109/TSA.2005.858005.

Emmanuel Vincent. Improved Perceptual Metrics for the Evaluation of Audio Source Separation. In *Latent Variable Analysis and Signal Separation*, pages 430–437. Springer, 2012. ISBN 978-3-642-28551-6.

Richard Vogl, Matthias Dorfer, Gerhard Widmer, and Peter Knees. Drum transcription via joint beat and drum modeling using convolutional recurrent neural networks. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 150–157, 2017.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *CoRR*, abs/1905.00537, 2019.

DeLiang Wang and Jitong Chen. Supervised speech separation based on deep learning: An overview. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(10):1702–1726, 2018.

Laurenz Wiskott and Terrence J. Sejnowski. Slow Feature Analysis: Unsupervised Learning of Invariances. *Neural Computation*, 14(4):715–770, 2002. ISSN 0899-7667, 1530-888X. DOI: 10.1162/089976602317318938.

Adrien Ycart and Emmanouil Benetos. A study on LSTM networks for polyphonic music sequence modelling. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 421–427. ISMIR, 2017.

Adrien Ycart and Emmanouil Benetos. Polyphonic Music Sequence Transduction with Meter-Constrained LSTM Networks. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 386–390. IEEE, April 2018. ISBN 978-1-5386-4658-8. DOI: 10.1109/ICASSP.2018.8462128.

Adrien Ycart, Daniel Stoller, and Emmanouil Benetos. A comparative study of neural models for polyphonic music sequence transduction. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 470–477, 2019.

Jinsung Yoon, James Jordon, and Mihaela van der Schaar. GAIN: Missing Data Imputation using Generative Adversarial Nets. *CoRR*, abs/1806.02920, 2018.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In *AAAI Conference on Artificial Intelligence*, pages 2852–2858, 2017.

Ning Zhang, Junchi Yan, and Yu Chen Zhou. Unsupervised Audio Source Separation via Spectrum Energy Preserved Wasserstein Learning. *CoRR*, abs/1711.04121, 2017.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, pages 2223–2232, 2017.