BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

MODELLING MUSIC WAVEFORMS USING WAVENET

MODELOVANIE HUDBY NA ÚROVNI SIGNÁLU POMOCOU WAVENETU

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **TERÉZIA SLANINÁKOVÁ**
AUTOR PRÁCE

SUPERVISOR **Ing. KAREL BENEŠ**
VEDOUCÍ PRÁCE

BRNO 2018

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Slanináková Terézia**
Obor: Informační technologie
Téma: **Modelování hudby na úrovni signálu pomocí WaveNetu**
Modelling Music Waveforms Using Wavenet
Kategorie: Zpracování signálů

Pokyny:

1. Seznamte se architekturou WaveNet
2. Implementujte ji ve vhodném toolkitu pro strojové učení
3. Natrénujte model na vhodné datové sadě
4. Z natrénovaného modelu vygenerujte syntetická data
5. Zhodnoťte kvalitu generované hudby
6. Vytvořte plakát a/nebo video shrnující výsledky práce

Literatura:

- van den Oord, A et al.: WaveNet: A Generative Model for Raw Audio

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2, a rozpracování bodu 3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Beneš Karel, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2
L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstract

This thesis focuses on exploring the possibilities of modelling music and speech with WaveNet, a deep neural network for generating raw audio waveforms. Using existing implementations, WaveNet was trained on multiple datasets and produced several audio files. Multiple experiments were carried out with various hyperparameter setups of WaveNet to find the optimal settings for the best results. Furthermore, multiple generation schemes were used, each having varying impact on the quality of generated audio. This quality was evaluated using human assessment via a questionnaire, where the musical samples were rated with a score 2–3.1818 on a 5 point scale, which is comparable to the rating of referential audio from the original WaveNet paper (3.1818).

Abstrakt

Práca sa zaoberá skúmaním možnosti modelovania hudby a reči pomocou WaveNetu, hlbokou neurónovou sieťou pre generovanie zvuku na úrovni signálu. Za pomoci existujúcich implementácií bol WaveNet netrénovaný na rôznych datasetoch a vyprodukoval mnohé zvukové súbory. Bolo vykonaných niekoľko experimentov s rôznym nastavením hyperparametrov WaveNetu. Taktiež bolo použitých niekoľko schém generovania, každá s rôznym vplyvom na generovaný výsledok. Kvalita výstupných zvukových súborov bola ohodnotená na základe dotazníku. Hudobné zvukové stopy dosiahli skóre 2–3.1818 na 5-bodovej škále, čo je porovnateľné s hudobnými nahrávkami originálneho výskumného tímu (3.1818).

Keywords

WaveNet, deep neural networks, music generation

Klíčová slova

WaveNet, hlboké neurónové siete, generovanie hudby

Reference

SLANINÁKOVÁ, Terézia. *Modelling Music Waveforms Using Wavenet.* Brno, 2018. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Karel Beneš

Rozšířený abstrakt

Bakalárska práca skúma možnosť generovania hudby pomocou WaveNetu. WaveNet je hlboká neurónová sieť umožňujúca modelovanie zvuku na úrovni signálu. Pôvodne bola predstavená v roku 2016 výskumným tímom DeepMindu. Po dosiahnutí predsvedčivých výsledkov v generovaní reči a autori okrajovo zamerali na hudbu. Cieľom tejto bakalárskej práce je preskúmať sa primárne na generovanie hudby, preskúmať viaceré výsledky vyprodukované viacerými nastaveniami WaveNetu a zhrnúť zistenia pre možné budúce snahy v tejto oblasti.

V úvode je rozoberaná problematika neruónových a hlbokých neurónových sietí, ich štruktúra a rozdelenie. Špeciálna pozornosť sa kladie konvolučným neurónovým sieťam, na ktorých je WaveNet sčasti založený. Následne analyzujem architektúru WaveNetu, generatívne a autoregresívne modely, keďže jedným z nich je aj WaveNet.

Následne sú opísané časté problémy pri trénovaní hlbokých neurónových sietí, ako napríklad pretrénovanie. V texte som načrtla rôzne riešenia tohto problému a niektoré z nich sú použité v experimentačnej fáze.

Na experimenty som použila existujúce implementácie WaveNetu, upravila som ich podľa potrieb tejto bakalárskej práce a vytvorila súbor skriptov pre ich jednoduché púštanie. Trénovanie bolo vykonané na rôznych voľne prístupných datasetoch a výsledky boli medzi sebou porovnané. Datasety boli rôznej kvality a veľkosti, vzorkované na frekvenciách od 8 kHz do 48 kHz. Najčastejšie zastúpený hudobný nástroj v hudobných datasetoch bol klavír, ale boli vykonané aj experimenty s gitarou, husľami, či flautou.

V rámci trénovania som preskúmala možnosti nastavenia rôznych hyperparametrov a ich vplyv na kvalitu výsledku. Príkladmi sú veľkosť WaveNetu, rôzna vzorkovacia frekvencia vstupných dát, rôzne nastavenie dropoutu, ako jednej z techník regularizácie, nastavenie rôznych hodnôt learning rate a pod. Ako výstup experimentov zahŕňam v texte graf trénovania s tréningovou a validačnou chybou, vyprodukované audio súbory vykresľujem pomocou grafov amplitúd v čase a spektrogramov. Ďalšie pomocné merítka kvality zahŕňajú rozloženie prevdepodobností amplitúd, ktoré vyprodukuje WaveNet, alebo entropia, či cross-entropia výstupu.

V texte ďalej rozoberám predprípravu trénovacích dát ich transformáciou tzv. μ-law algoritmom do 8-bitového priestoru.

Na generovanie bola použitá vylepšená verzia pôvodného generovacieho procesu, ktorá využíva ukladanie si už vypočítaných medzi-výsledkov a tak predchádza zbytočným výpočtom a zrýchľuje celý proces. Skúšala som taktiež viaceré schémy generovania s rôznou obtiažnosťou. Prvým bol tzv. teacher-forcing, kde sa ako vstup generovania každej vzorky používa referenčný audio súbor. Kvalita výsledkov tu bola naväčšia, aj podtrénovaný model zvládol kvalitne replikovať referenčný súbor. Druhým prístupom bola inicializácia generovacieho procesu referenčným zvukovým súborom. Tento prístup produkoval zvuky mierne horšej kvality, ale pri dostatočnom natrénovaní bolo v jeho výsledkoch možné rezoznať isté hudobné prvky. Nakoniec som vyskúšala najobtiažnejšiu schému – unikátne generovanie bez referenčnej nahrávky. Táto schéma bola pre WaveNet najobtiažnejšia, výsledky boli často veľmi zašumené, alebo naopak tiché.

Pre ohodnotenie kvality výstupov z experimentov som použila dotazník, ktorý bol vyplnený jedenástimi respondentmi. Za použitia tzv. MOS techniky boli urobené závery a finálne porovnanie vyprodukovaných zvukových stôp s nahrávkami od pôvodného výskumného tímu. Záverom z dotazníka bolo porovnanie kvalitatívnych hodnôt nahrávok a vyvodenie záverov. Jedna hudobná nahrávka získala rovnaké skóre (3.18), ako zahrnutá referenčná

nahrávka z pôvodného výskumného tímu. Ostatné boli hodnotené mierne horšie (skóre 2 z 5). V prílohe prezentujem rozloženie odpovedí na jednotlivé otázky dotazníka.

V závere sú zhrnuté výsledky, prezentované problémy, ktoré sa pri práci vyskytli, ich príčiny a možné riešenia.

Všetky vygenerované zvukové súbory, ako aj zdrojové kódy sú uložené v mojom github repozitári[1].

[1] https://github.com/TerkaSlaninakova/BP

Modelling Music Waveforms Using Wavenet

Declaration

Hereby I declare that this bachelor's thesis was prepared as an original author's work under the supervision of Ing. Karel Beneš. All the relevant information sources that were used during the preparation of this thesis, are properly cited and included in the list of references.

.....
Terézia Slanináková
May 16, 2018

Contents

Chapter 1

Introduction

Music is a ubiquitous artistic activity, that was found in every human culture and may have been in existence for at least 55,000 years [43]. Many people, often without any musical background, experience intense emotions in reaction to music, that stem from activations in brain regions connected to euphoric reward responses [6].

The most dominant features of music are *pitch*, representing the sound's position on a frequency-related scale, *rhythm*, *dynamics* – the variation in loudness and *timbre*, representing the distinctive sound quality. Musical composition, the process of creating unique pieces of music, makes use of relationships between these features. For example an interval, the difference between two distinctive pitches, is a basic consideration in melody composition.

Algorithmic music generation (also known as *algorithmic composition*) is the process of creating music without human intervention.

Systems capable of generating music are divided according to the underlying theoretical foundation. For example *Grammars* [31] examine music similarly as a language, i.e. using textual descriptions of its characteristics, such as harmonies and rhythm.

On the other hand, in *Mathematical systems* [14], the composition is controlled as a stochastic process – by the use of non-deterministic methods and random events. First attempts that used formal mathematical means date back to 1958 Xenakis' compositions of *Analogique A et B* [45] using Markov chains. Models based on Markov chains suffered mainly from modeling the music as a plain sequence of notes, while musical pieces are usually structured as progressions of long-scale bars.

Machine learning based techniques, such as *Recurrent Neural Networks* (RNNs), are better in describing the long-term characteristics of music [15]. Particularly well suited for this task are RNNs with *Long Short-Term Memory* (LSTM) units, which are more precise in capturing long-term temporal dependencies [12].

Recently, Convolutional Neural Networks, traditionally working with visual data, were successfully applied to audio generation as well. The goal of this thesis is to use WaveNet [34], a generative model for raw audio based on Convolutional Neural Networks, to generate unique sequences of sound.

Chapter 2

Neural Networks

Neural networks (more specifically *artificial neural networks*) are computing systems loosely inspired by biological neural networks designed to perform certain tasks without explicit programming. Their basic unit is a *neuron*, visualized in Fig. 2.1, which receives an input from a neuron or series of neurons, performs some kind of transformation on it and passes it to the next set of neuron.

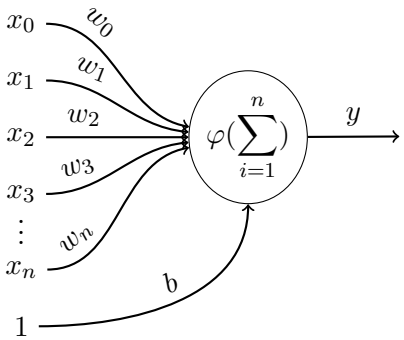


Figure 2.1: A single neuron with input in the form of vector $\vec{x}$ of length n. The input vector has a weight vector $\vec{w}$ of the same length associated with it. Additionally, there is a bias term 1 with implicit weight of b.

The weights are real numbers expressing the importance of the respective input to the output and the bias term introduces a trainable constant value to the expression. Neuron's output is computed as a weighted sum of the input values:

$$y = \varphi(\sum_{i=1}^{n} x_i \cdot w_i + b) = \varphi(\vec{w}^T \vec{x} + \vec{b})$$

Result of the weighted sum is oftentimes further transformed by an *activation function* denoted as φ. It is used to transform the activation level of a neuron into an output signal [22]. Although a large enough neural network using any activation function can approximate arbitrarily complex functions [10], ultimately, the choice of an activation function affects backpropagation and therefore the training process as a whole [29].

There are several types of activation functions, some outperforming others, e.g. *Sigmoid*, *ReLU* or *Tanh*.

Sigmoid is historically the most used one. It has the intuitive property of either not propagating neuron at all or propagating it at maximum frequency, see Fig. 2.2. It is

scarcely used today though, mainly because of its two drawbacks: close to zero gradients for most input values and not having zero-centered activations. Gradients nearing zero occur when neuron's activation is close to zero or one. Then the gradient will be almost zero, meaning its value will not be regarded during backpropagation.

Having activations, that are not zero-centered is problematic, because gradient on the weights will in subsequent layers become either all positive or all negative, which could negatively affect the dynamics in the updates of gradient for the weights (so-called *zig-zagging*) [29].

Tanh is similar to Sigmoid, with the same drawback of disregarding gradients of activations that are nearing -1 or 1, but it is zero-centered, so it is usually preferred over Sigmoid.

Rectified Linear Unit (ReLU) is a simple, see Fig. 2.2, yet popular activation function, which has shown faster convergence of stochastic gradient descent compared to Sigmoid or Tanh [25]. One downside, stemming from ReLU assiging hard zero for negative values, is that the affected neurons cannot be re-activated during training, which results in so-called „dying ReLU" [27]. Several variants of ReLU were introduced that attempt to solve the *dying ReLU* problem, e.g. *Leaky ReLU* [30]. This problem can be also mitigated by initializing ReLU neurons with slightly positive biases.

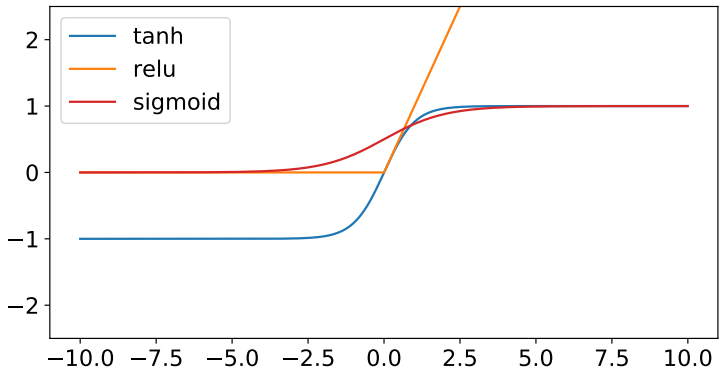


Figure 2.2: Comparison of different activation functions. Sigmoid and Tanh are examples of *saturating* activation functions, meaning they compress any real-numbered input into a fixed limited range: $[0, 1]$ and $[-1, 1]$ respectively. ReLU is not bounded above.

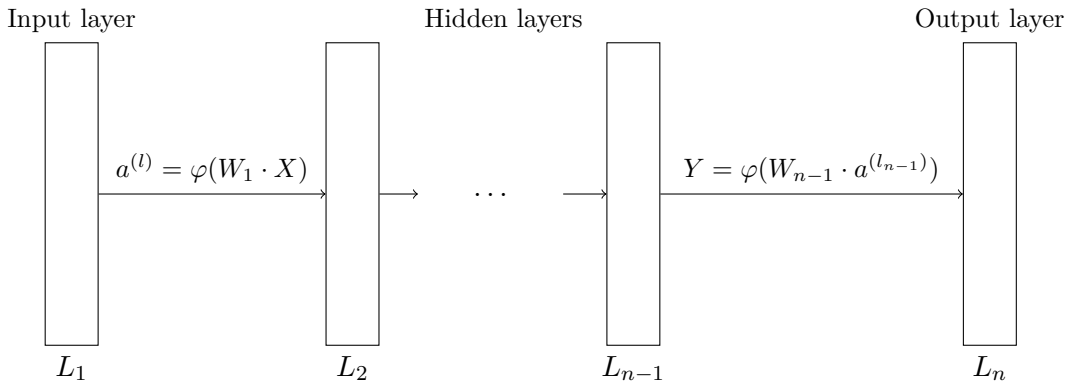


Figure 2.3: Interconnected neurons are organized into layers. For each layer, weight matrix is applied to the output of the previous layer, bias vector, which is omitted from the Figure for brevity, is added and the final expression is fed into the activation function.

Goups of cooperating neurons are structured into layers to form *neural networks*. Fig 2.3 shows a *Feedforward* Neural Network. A neural network is considered to be feedforward if the connections between neurons do not form a cycle, as opposed to *Recurrent* Neural Networks, that do have cycles. If there are many hidden layers, a neural network is called *deep*. The exact number is not clearly established, but usually having more than two hidden layers counts as *„deep“*. In contrast to Deep Neural Networks, we refer to networks with less hidden layers as being *shallow*.

Deep neural networks began to be widely used only in approximately the last 10 years and have dominated various machine learning competitions since. In 2006, Hinton et al. published one of the earlier breakthrough papers [20] demonstrating an efficient way to train deep networks with the use of unsupervised pre-training, that had just a single layer of feature-detecting units. Today, the most successfull networks are much deeper, e.g. ResNet [18], containing up to 152 layers.

Generally speaking, even though a shallow neural network can approximate any function, i.e. can in principle learn any mapping [4], depth can lead to an exponential reduction in the number of neurons required, for specific functions or specific neural networks [13].

2.1 Convolutional Neural Networks

The focus of this thesis is WaveNet's architecture, which is a variant of Convolutional Neural Network (CNN). CNNs are mainly used in areas of image recognition and classification, but have been also shown useful in the fields of speech recognition [46] or text classification [23]. Their central operation is convolution, which is a way of feature extraction from the input data. Convolution involves sliding a matrix (= *filter* or *feature detector*) over the input and computing the dot product to create an *activation map*. In practice, there are usually multiple filters that are independently applied over the input to produce equally many *feature maps*. The number of feature maps denotes the depth of a convolution layer – not to be confused with depth of a neural network discussed above. An activation function is then applied over the set of feature maps to introduce non-linearity. This process is visualized in Fig. 2.4.

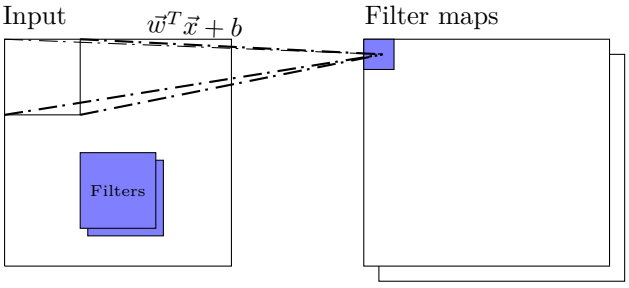


Figure 2.4: The convolution operation. A filter is placed at a particular location over the input matrix, its components are multiplied (dot product) to produce a single scalar for every component. The filter is then slid by a fixed number of elements over the input and this operation is repeated, thus obtaining the filter map.

A layer frequently appearing in Convolutional Neural Networks is a *pooling layer*. Pooling layer is inserted in between two successive convolution layers. It works by moving window (a *stride*) across the input space, combining multiple neurons into one, that will be propagated to the next layer. Pooling is used to reduce the dimensionality, spatial size and the number of parameters of the input. It also solves the *overfitting problem* described in Section 2.2.

Except for the last one or two layers, only a small portion of neurons in one layer is connected to neurons in the next layer, extent of this connectivity is referred to as *Receptive field* of the input feature. An overview of an example CNN architecture is visualized in Fig. 2.5.

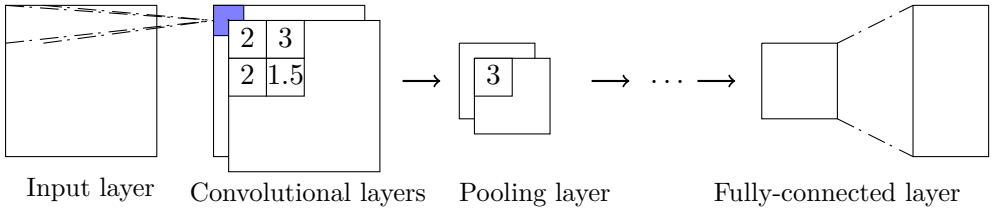


Figure 2.5: The CNN architecture. The last layer is *Fully-connected*, meaning that it contains neurons with full connections to activations in the previous layer. „...“ denotes the repetition of convolution and pooling layers.

2.2 Regularization techniques

Overfitting is one of the biggest problems in training neural networks [39]. It occurs when a model performs well on the training data, but fails to generalize enough to perform well on validation data as well. This happens because there are accidental regularities introduced with the training data. The trained model then cannot differentiate between the general regularities that should be learned and those which are caused by sampling error.

Multiple techniques were developed to tackle this problem, such as *noise injection*, *weight decay*, special cross-validation variants, *early stopping* or *dropout* [38].

Noise injection tackles overfitting by penalizing complex models indirectly by adding noise to the training dataset [40]. Weight decay adds a regularization term to the network's loss to compute the backpropagation gradient [26].

In an early stopping setup, the dataset is split into a training set and a validation set. Validation set is used as an indicative of overfitting during training. The training procedure is stopped, just before the weights have converged. This convergence is determined by various markers, such as validation loss being higher than it was the last time it was checked, or so-called *patience*, i.e. the number of epochs to wait before early stop if there is no progress on the validation set [9].

Dropout [41] is a simple regularization technique, which randomly drops certain percentage of neurons from the neural network during training, as visualized in Fig. 2.6. For each training case in a mini-batch, the dropping of neurons results in creation of a „thinned“ neural network. Forward and backpropagation for that training case are done only on this thinned network. This prevents neurons from co-adapting too much on the training data [41] .

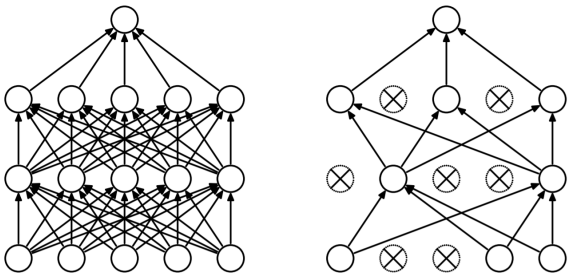


Figure 2.6: Left: A Neural Network with 2 hidden layers. Right: A Neural Network with dropout (dropped units are crossed). Taken from [41].

2.3 Autoregressive generative models

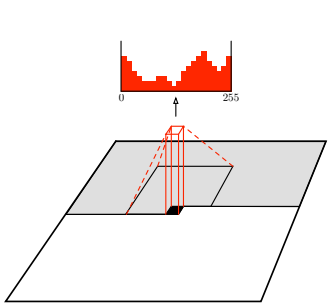
Generative models are a part of unsupervised machine learning applications. They model the generated data in a probabilistic fashion, learning the joint probability distribution over a data point and target (label) values.

Generative models are often contrasted with *discriminative models*, that directly model the conditional distribution $p(y|x)$ without taking the input distribution into account [8]. The goal of generative modeling is to discover relationships between parts of the data. Examples of neural generative models include Generative Adverserial Networks [17], Variational AutoEncoders [24] or autoregressive models.

Autoregressive models represent a class of models that use values from the previous time step to predict future ones, exploiting the fact that time series data is inherently sequential:

$$p(x) = \prod_{t=1}^{T} p(x_t|x_1, ..., x_{t-1}) \tag{2.1}$$

The nature of Autoregressive models was used by models such as PixelCNN [35] to model images pixel by pixel. The use of autoregression is seen in Fig. 2.7.



1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0

Figure 2.7: Prediction of the next pixel from a neighbourhood of previous ones. Right: A matrix used to mask 5x5 filters, to guarantee that the neural network is not influenced by the pixels below in order to preserve the conditional probability of the data. Visualization from PixelCNN [35].

Chapter 3

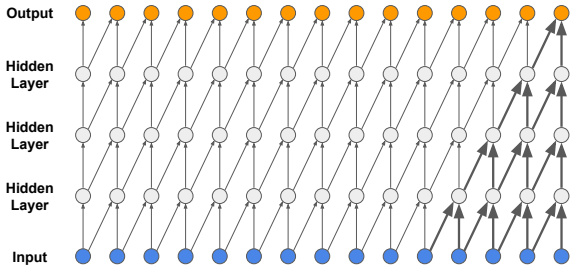
Wavenet

WaveNet, the focus of this thesis, is a direct successor of PixelCNN and its extension to the audio domain. Wavenet [34] is a deep neural network for generating raw audio waveforms. In the original paper, van den Oord et al. demonstrate a state-of-the-art performance when applied to text-to-speech problems. Furthermore, the model can capture characteristics from multiple speakers and can switch between them after being conditioned on the identity of the speaker. The same network can be also used to synthesize other types of audio input, such as music. Since it was published in September 2016, the architecture of WaveNet saw several improvements: the original research team managed to achieve a 1,000-fold performance improvement and launch it in Google Assistant [36]. Researchers from Baidu created a model based on Wavenet with faster training time [3].

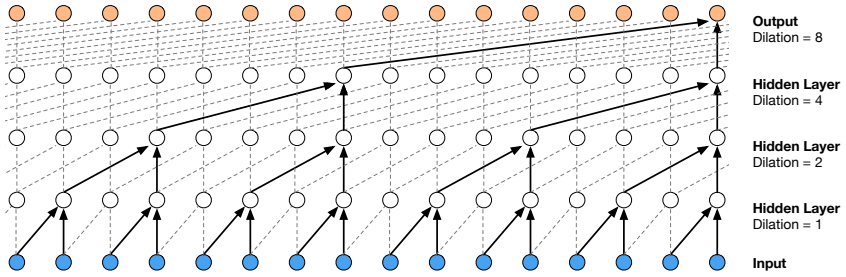
3.1 Architecture

Wavenet is an autoregressive model, that operates directly on the audio waveform level by modelling the output sample by sample.

The key element of its architecture are dilated causal convolution layers. The word 'causal' simply implies that the filter output does not depend on any future inputs and therefore the conditional probability of the modeled data is kept. Dilated causal convolutions (also known as convolutions à trous, or convolutions with holes) were introduced to increase the receptive field at a reasonable computational cost, see Fig. 3.1.



(a) A stack of causal convolutional layers where k-th neuron in layer l is influenced by neurons k and $k-1$ from $l-1$. The receptive field of size 6 in this case denotes how many neurons in the input layer influence the output neuron. Taken from van den Oord's talk at SANE 2017 [32].

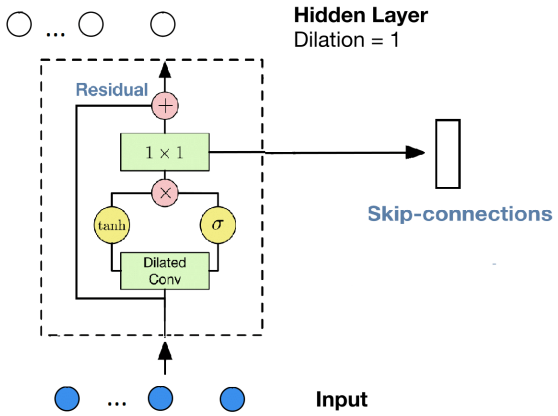


(b) A stack of dilated convolutional layers where an increase of receptive field was achieved by skipping over input values with a certain step. Increasing the skip step on every layer then allows for exponential, rather than linear increase of receptive field. Taken from [33].

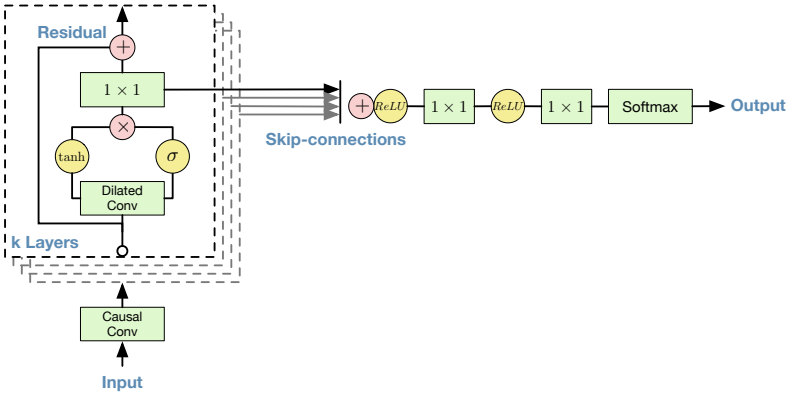
Figure 3.1: To predict a sample, the generating process has to take thousands of previous samples into account, which is computationally costly. To be able to produce receptive field of this magnitude, the dilations are doubled for every layer up to a limit and then repeated.

In order to speed up the training convergence, WaveNet makes use of *residual connections* [18] throughout the network. Residual connections try to mitigate some of the problems in training deep neural networks. Such as the *vanishing gradient problem*, which arises during backpropagation, when a weight receives vanishingly small gradient preventing it from changing its value. Another one is *degradation* [19], denoting the observation, that deeper neural networks are more difficult to train than shallow ones, due to increase in accuracy saturation. Intuitively it can be thought of as valuable information not getting passed through too many layers of the network. Residual connections solve it by introducing „shortcut“ connections, that skip one or more layers.

In WaveNet, residual and skip connections are integrated into every layer, with the output of residual connection providing input into the next layer and output of skip connection being stored for every layer and post-processed once the final layer is computed, see Fig. 3.2.



(a) A single residual block between a layer of inputs and the first dilation layer, adding to the abstraction visualized in 3.1b.



(b) The WaveNet architecture in expressed in terms of residual blocks. Residual and skip connections are evaluated for every layer, resulting in a collection of skip-connections, one from every dilation layer. A series of post-processing transformations is done to produce the softmax distribution of samples.

Figure 3.2: Visualization of the residual blocks' position in WaveNet's architecture.

A waveform $x = x_1, ..., x_T$ is represented by a joint probability that is factorized as s product of conditional probabilities, as expressed by Eq. 2.1. The conditional probability distribution is modeled by a stack of causal convolutional layers with using Softmax in the output layer, see Fig. 3.2. There are no pooling layers in the network.

As an activation function, WaveNet uses gated activation units:

$$z = \tanh(W_{f,k} * x) \odot \sigma(W_{g,k} * x)$$

Where the convolution operator is denoted by *, element-wise multiplication by $\odot$, σ is the logistic sigmoid function, k is the layer index, f and g filter and gate and W is the learnable filter used in convolution.

3.2 Original results

In the original paper, van den Oord et al. experimented with WaveNet on three different tasks: free-speech generation (not conditioned on text), speech conditioned on text (text-to-speech), and music audio modeling. Produced audio files are available in the Deepmind blog introducing Wavenet[1].

Free-form speech generation used the VCTK corpus [47]. The model generated non-existent but human-like language with realistic intonations. Additionally, Wavenet was able to capture characteristics of a single speaker by conditioning it on one-hot encoding of the speaker's identity.

For speech synthesis experiments, North American English dataset and Mandarin Chinese dataset were used. During training, Wavenet was conditioned on linguistic features derived from the input text and logarithmic fundamental frequency. As a contrastive system, HMM-driven unit selection concatenative [16] and LSTM-RNN based statistical parametric [48] synthesizers were built. Evaluation was conducted by the mean opinion score (MOS) tests, where paired with comparison tests, subjects were asked to choose which sample they preferred. The results showed that WaveNet outperformed the baseline statistical parametric and concatenative speech synthesizers in both English and Mandarin.

For music modeling, the MagnaTagATune and YouTube piano datasets, described in Chapter 5, were used. Generated samples were harmonic and musical, especially after enlarging the receptive field from several miliseconds, as used for speech to several seconds [34].

[1] https://deepmind.com/blog/wavenet-generative-model-raw-audio/

Chapter 4

Codebase description

WaveNet's source code was not published with the original paper and several specifics of the network's implementation were missing. Despite this, various implementations appeared on github[1,2], focusing not only on audio, but image[3] and text generation[4] as well. I decided to experiment with an implementation WaveNet in Python3 using a Deep Learning framework TensorFlow [1].

I did not to write a WaveNet implementation from scratch, instead I reused some existing concepts and merged them into one solution. The implementation used in experiments is partially based on tensorflow-wavenet[1] and generation scheme on fast-wavenet [37]. Sequence diagram of the whole program is shown in Fig. 4.1.

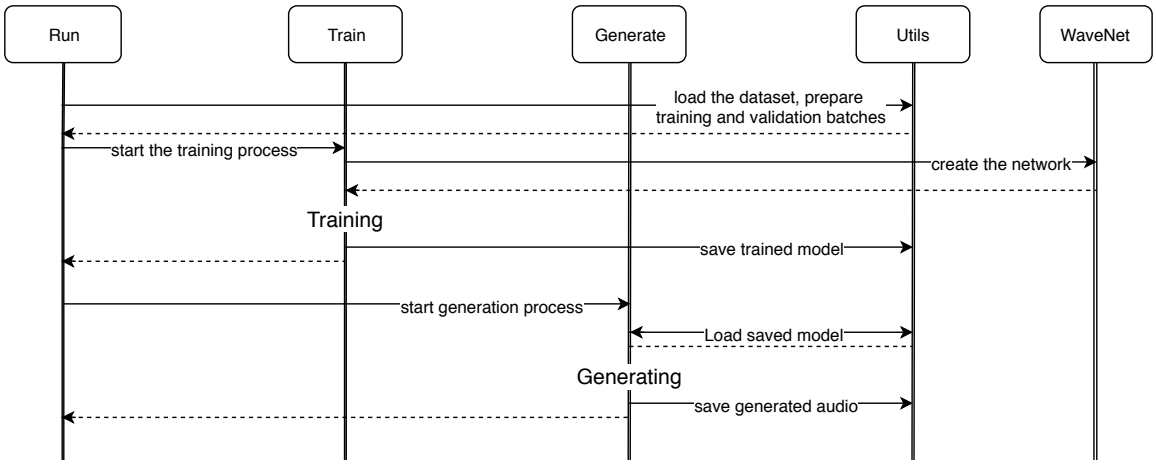


Figure 4.1: Overview of the entire codebase. *Run* is orchestrating the execution, *Train* is used for pre-processing the dataset and handling the training process. *WaveNet*, largely based on 1, creates WaveNet and *Generate* contains implementation of the generative process using Fast-Wavenet [37]. *Utils* are a collection of supporting actions, such as audio loading and writing, constructing training batches, logging, plotting graphs and preparing the environment.

[1] https://github.com/ibab/tensorflow-wavenet
[2] https://github.com/basveeling/wavenet
[3] https://github.com/Zeta36/tensorflow-image-wavenet
[4] https://github.com/Zeta36/tensorflow-tex-wavenet

The execution starts with loading of the dataset with desired sampling rate. I used variation of sampling rates, ranging from 8 kHz to 48 kHz. Pre-processing begins with a μ-law transformation as described in Section and continues with one-hot encoding of the sample to allow the model to learn an embedding for each discrete input value.

After the audio loading and pre-processing phase, training and validation batches are created randomly from the dataset, the ratio being 80:20 in favor of training data. I approach training in terms of epochs, one epoch being iteration of the training through all the training data. After every epoch the order of training data is shuffled on the recording level to prevent overfitting and help the model remain general [5].

The generation process, described in Section 4.3, produces a vector of probabilities, each corresponding to an amplitude bin of the audio in 8-bit space (256 values). The post-processing phase is then employed, involving the use of a transformation inverse to μ-law to get the floating-point amplitudes of the resulting waveform.

The whole execution is adjustable, parameters of the run are defined through *parameters.json* configuration file. Thanks to Tensorflow's Save and Restore functionality[5], the trained model is being saved throughout the training process, so that it can be later use to resume the training or initialize the generation process. One execution process can therefore involve only generating the output audio from an existing model.

The codebase is available in my github repository[6] together with the experiments.

4.1 Audio pre-processing

In order to adjust the dataset before entering the training phase, two transformations were used: *audio segmentation* and *μ-law transformation*.

Due to limitations of the training environment, It was not possible to run WaveNet on files of a song length, i.e. over 3 minutes. Some of the datasets, e.g. MagnaTagaTune and Youtube-8M described in Chapter 5, provide only such audio files. In order not to disregard them just for this limitation, I segmented the downloaded datasets into parts of 8 seconds for Youtube-8M and 2 seconds for MagnaTagATune. In a 8 kHz and 16 kHz sampling rates, which are used for Youtube-8M and MagnaTagATune respectively, that amounts to 64000 and 32000 samples per one audio file.

4.1.1 μ-law

An audio compression scheme, known as the μ-law transformation, is used to pre-process input from all the datasets. This transformation is often used in analog signal transmission with the goal of reducing wide dynamic range of an input audio signal [11]. With the input $x; x \in [-1, 1]$, the equation for μ-law encoding is the following:

$$f(x_t) = \text{sgn}(x_t)\frac{\ln(1+\mu|x_t|)}{\ln(1+\mu)}$$

where $\mu = 255$ (8 bits).

The reverse process of decoding is given by the following equation:

$$f^{-1}(y) = \text{sgn}(y)\frac{1}{\mu}((1+\mu)^{|y|} - 1) \qquad -1 \leq y \leq 1$$

[5] https://www.tensorflow.org/programmers_guide/saved_model

[6] https://github.com/TerkaSlaninakova/BP

4.2 WaveNet

Implementation of WaveNet closely follows the overview of the architecture in Fig. 3.2. The entry point to the network for the pre-processed input is a single convolutional layer, which reduces the channel dimension. It is followed by multiple stacks of dilated convolutional layers. Their exact number is modifiable by two hyperparameters: number of dilations and number of blocks, as shown in Fig. 4.2.

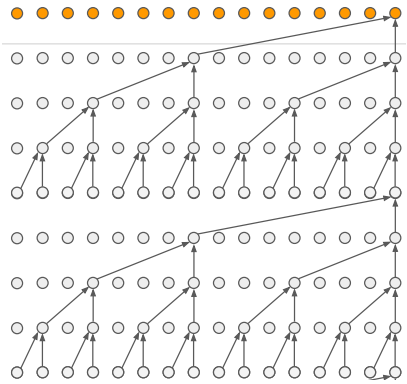


Figure 4.2: There can be multiple stacks of dilations of the same dilation factor. This visualization shows 2 blocks with exponentially increasing dilation factor of 1, 2, 4 and 8. Taken from van den Oord's talk at SANE 2017 [32].

Dilation layers use skip and residual connections, as visualized in 3.1b.

The outputs of all the layers, propagated by skip-connections, are transformed back to the original number of channels using two post-processing layers, as visualized in Fig. 3.2. These are followed by a softmax, that produces an output in the form of categorical distribution over the quantization levels. Once whole waveform is predicted, it is converted into a floating-point sequence of amplitudes by the reverse μ-law transformation scheme.

4.3 Generation process

The sequential nature of creating the waveform makes the generation process computationally expensive. Expressed in terms of algorithmic complexity, the overall computation time for a single output is $O(2^L)$, L being the number of layers in the network [37]. The training phase is faster, since conditional predictions can be made in parallel. Paine et al. presented [37] an improvement over the original WaveNet's generation, called Fast-Wavenet.

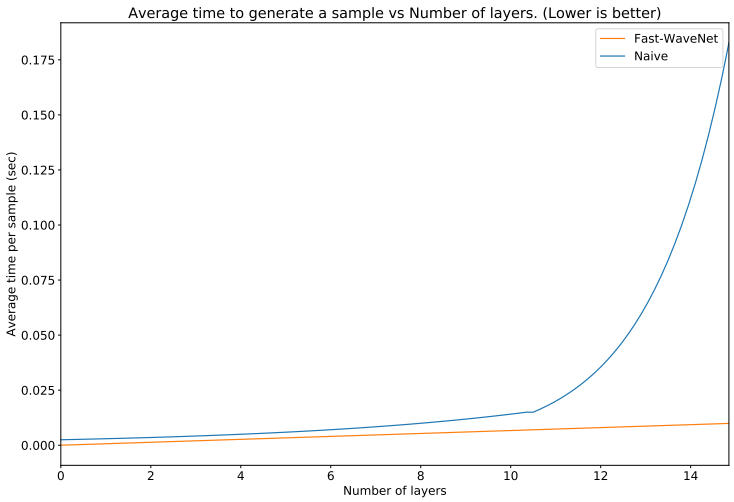
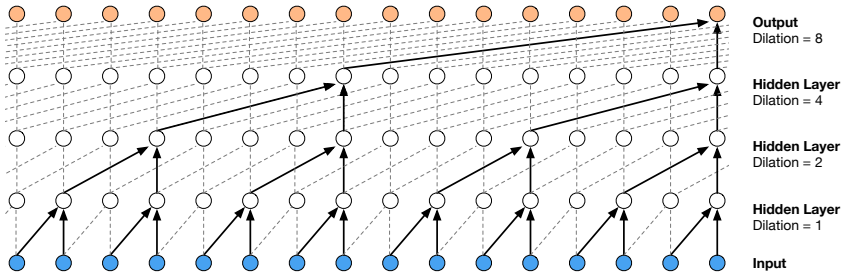


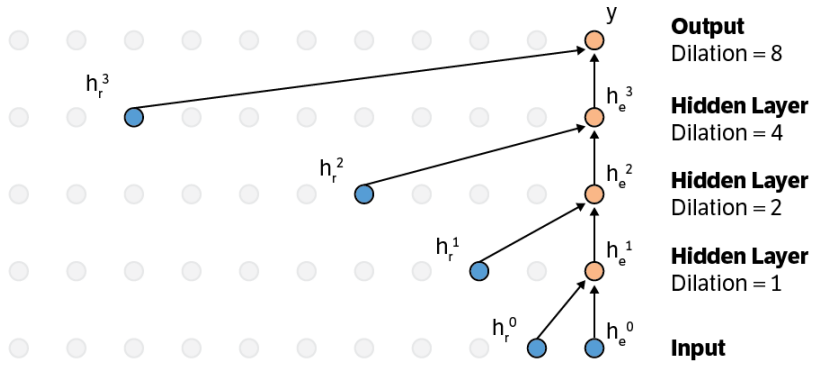
Figure 4.3: Fast-Wavenet works by reducing the generation cost from $O(2^L)$ to $O(L)$, L being the number of layers. Timing experiments performed in [37].

The basic premise of fast-wavenet is to view the computational graph needed to get a single output as a binary tree. Since the model is being applied repeatedly, there is a lot of redundant computation, see Fig. 4.4. A single node in this computational graph is referred to as a *recurrent state*.

Because of dilated convolutions, a single output depends on recurrent states spanning several timesteps back, as opposed to just the immediate predecessors. Fast-WaveNet introduces FIFO queues, that hold the cached recurrent states for the respective layer.



(a) The naive generation process, taken from [34].



(b) The generation process introduced in fast-wavenet. h_e^0 is the input sample, h_r^0 the first recurrent state, h_r^3 the fourth one. Length of the FIFO queue holding recurrent states for a given dilation depends on the dilation factor: 2^0 for the first state, 2^3 for the fourth one. Taken from [37].

Figure 4.4: Comparison of the naive and improved generation process by Fast-WaveNet. The input sample is thought of as „embedding“, therefore is named as h_e^0. The recurrent states are referred to as h_r^n.

4.3.1 Generating a single sample

WaveNet outputs a distribution of probabilities out of which one value is chosen, representing one amplitude encoded using μ-law, as show in Fig. 4.5.

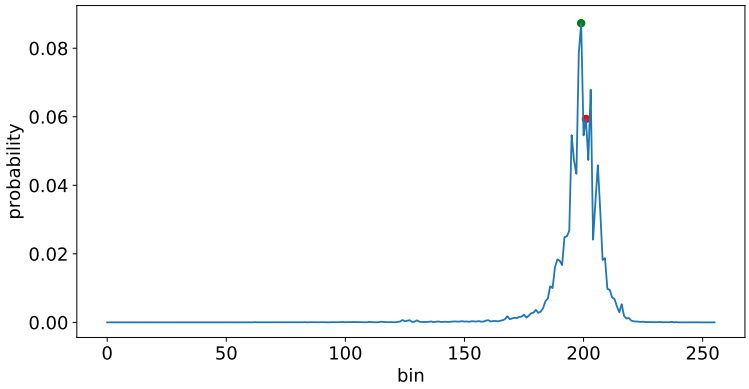


Figure 4.5: WaveNet's output in the form of probability distribution. This image captures generation using teacher forcing, with red dot symbolizing the ground truth and green dot representing the chosen sample.

I used two approaches of choosing a single sample. The first and most straight-forward one is to pick the one with the highest probability, represented by the green dot in Fig. 4.5. The second, sampling approach is to pick one value out of the distribution randomly, given the probabilities of the distribution. Choosing a sample with the highest probability works fine in teacher-forcing setup, but results in silence in seeding and unique data generation schemes. For these two generation schemes I decided to use the sampling approach.

4.3.2 Generation scheme

WaveNet is a generative model that can produce a unique audio without following specific targets, which was done in the original paper with speech as well as music. However, to fully test WaveNet's generating abilities, three generation schemes were used: Teacher forcing, seeding and unique generation.

Teacher forcing is a technique mostly seen in the context of Recurrent Neural Networks [44], in which the ground truth output $y(t)$ is used as an input for the model at time $t+1$.

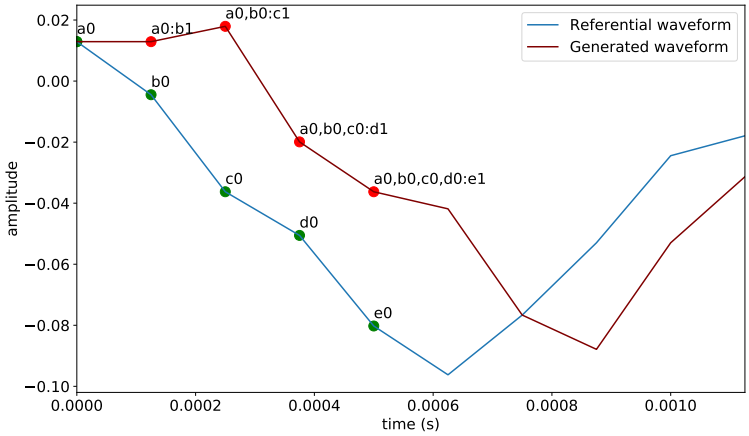


Figure 4.6: Visualization of teacher forcing. In this setup, the generation process uses the referential audio to produce a sample. Every sample is generated from its direct referential predecessor and a recurrent states created from a context of several previous samples, as explained in Section 4.3. For example the third sample of the generated waveform ($c1$) is given by $b0$ and $a0$. The first sample is assigned from the referential audio, because there is nothing to predict it from.

The second setting was using so-called *seeding*, i.e. providing a value from which the generation starts, based on the ground truth. Since every sample has to be generated from a previous one, the first sample has to be chosen outside of the generation scheme. In this case I used an existing audio to seed the generation with for the length of receptive field of the waveform.

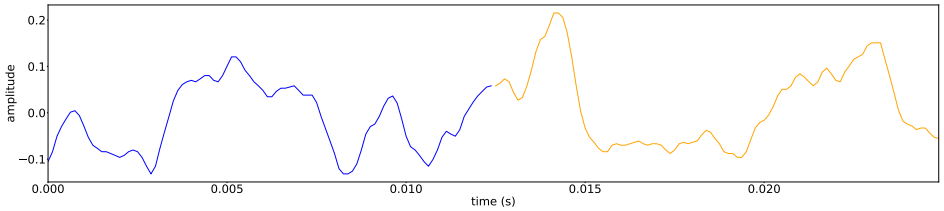


Figure 4.7: The seeding scheme. The waveform consists of the seeded part (an existing audio), shown in blue and the generated part (orange).

The last approach was to let WaveNet produce exclusively unique data without guiding it based on any other existing audio. The very first sample was chosen randomly, from the fixed range $[0, 2^8 - 1]$, corresponding to the 8-bit space.

Chapter 5

Datasets

In this chapter, the datasets used in the experiments are described. I used both speech and music datasets for training. In music, I used mostly focused on datasets with clean recordings of a single instrument to avoid overlaps of different sounds. For speech, I used the VCTK dataset as used in original WaveNet paper, described in section 5.1.

5.1 VCTK - The Voice Cloning Toolkit

VCTK [47] is an English multi-speaker corpus from the Centre for Speech Technology Research (CSTR). The dataset consists of 44 hours of audio recordings from 109 different speakers. Each speaker reads out about 400 sentences from newspaper plus the Rainbow Passage[1] and an elicitation paragraph[2] originally recorded with the intention of identifying the speaker's accent.

Speakers read different set of the newspaper sentences, that were selected using a greedy algorithm to maximize the contextual and phonetic coverage, while the Rainbow passage sentences were the same for all the speakers [42]. The recordings were originally made at 96 kHz sampling frequency and were later downsampled to 48 kHz. I used multiple sampling rates with this dataset ranging from 8 kHz to 48 kHz, the difference between them is apparent from Fig. 5.3. For an example of one audio clip, see Fig. 5.1.

[1] The Rainbow Passage can be found in the International Dialects of English Archive: (http://web.ku.edu/~idea/readings/rainbow.htm)

[2] The elicitation paragraph is as follows: *Please call Stella. Ask her to bring these things with her from the store: Six spoons of fresh snow peas, five thick slabs of blue cheese, and maybe a snack for her brother Bob. We also need a small plastic snake and a big toy frog for the kids. She can scoop these things into three red bags, and we will go meet her Wednesday at the train station.*

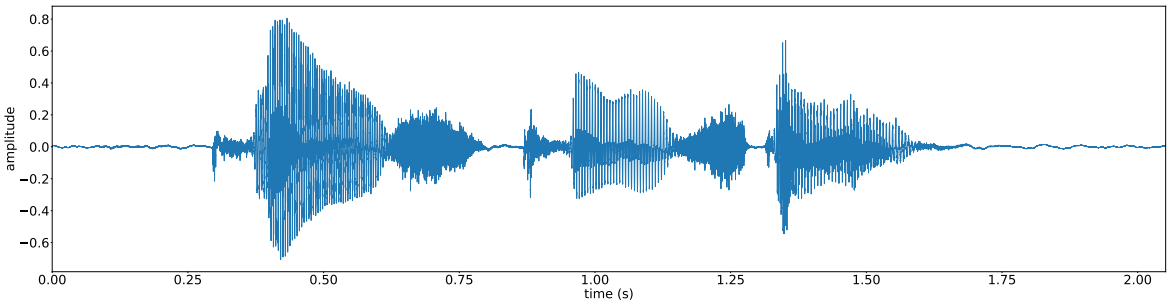
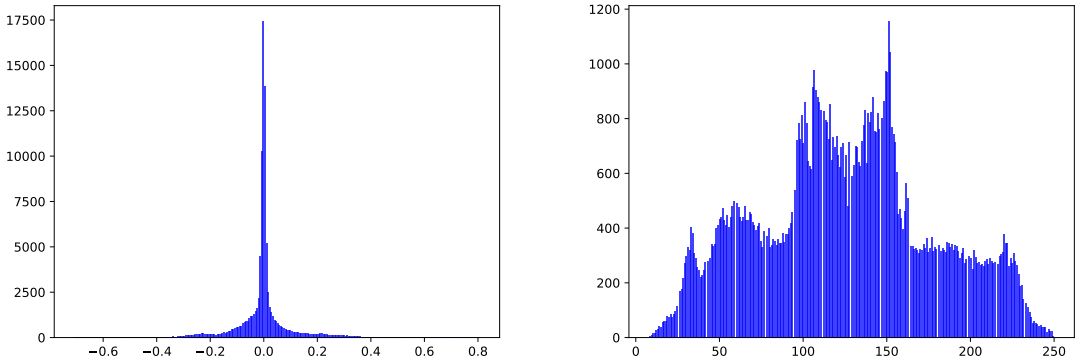


Figure 5.1: A raw waveform of a 23-year-old female from Southern England (speaker p225) saying „Please, call Stella“.

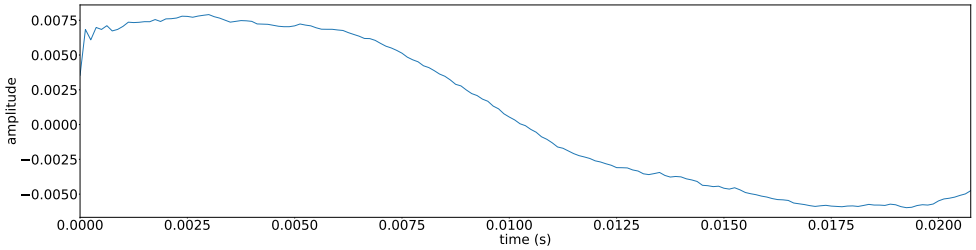
The motivation behind the use of μ-law, as described in Section 4.1.1, is apparent from Fig. 5.2.



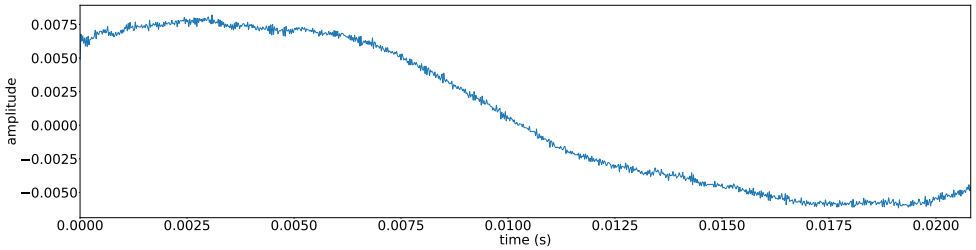
(a) Histogram of the original audio without pre-processing.

(b) Histogram of the same audio after a *mu*-law pre-processing.

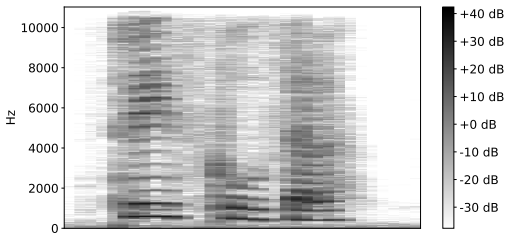
Figure 5.2: Amplitude histogram of the same audio as in Fig. 5.1, after a μ-law transformation used in the pre-processing phase, which adjusts the distribution to make it more uniform.



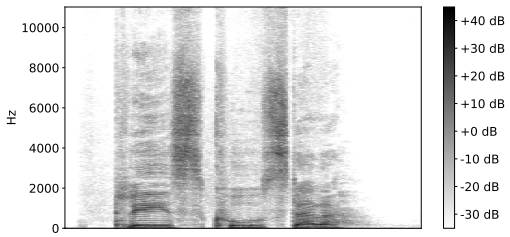
(a) First 21ms of audio loaded at 8 kHz sampling rate.



(b) Audio loaded at 48 kHz sampling rate.



(c) Spectrogram of a 8 kHz audio.



(d) Spectrogram of a 48 kHz audio.

Figure 5.3: An overview of the sampling rate's influence on the audio quality. Audio is the same as in Fig. 5.1.

5.2 IRMAS: a dataset for instrument recognition in musical audio signals

IRMAS is a dataset primarily collected for training and evaluating methods of instrument detection in audio files [7]. The included instruments are: cello, clarinet, flute, acoustic guitar, electric guitar, organ, piano, saxophone, trumpet, violin, and human singing voice. Additionally, there are further distinctions between individual genres the instruments are involved in, such as jazz, pop, rock of classical.

The dataset consists of 6705 audio files in 16 bit stereo sampled at 44.1 kHz. These are excerpts of 3 seconds from more than 2000 distinct recordings. With the focus on isolating specific instruments, in training I used only a subset of the dataset. For example for piano, there are 731 recordings that account for 36 minutes of data.

5.3 MagnaTagATune

MagnaTagATune is a dataset generated using a two-player online game TagATune [28].

This approach represents a shift from the conventional use of music tagging algorithms to annotate music. Traditionally, the evaluation metric of such algorithms are the measures of agreement between the output and the ground truth set. Law et al. presented a new evaluation approach consisting of collecting the judgments from players of the TagATune game. Given a human player's guess is denoted as $G = 0, 1$ and the ground truth is denoted as $GT = 0, 1$, the algorithm's performance metric defined as:

$$P_{i,j} = \frac{1}{N} \sum_{n=1}^{N} \sigma(G_{n,j} = GT_j)$$

where N is the number of players who saw the tags produced by an algorithm i on clip j, and $\sigma(G_{n,j} = GT_j)$ is a function which returns 1 if, for a sound clip j, the player n's guess and the ground truth equal.

The generated dataset consists of over 16 kHz-sampled 25000 audio clips of about 30 seconds each annotated with 188 tags taken from 5405 songs from the Magnatune label[3]. The musical clips are very variable, containing genres such as jazz, punk, folk, etc.

I used a subset of the dataset of audio clips labeled as 'piano'. Unfortunately, many of the recordings were not available at the stated url in the metadata csv[4]. Additional complication was an overlapping voice inserted by the Magnatune label commenting on the recording. Because of this, it was necessary to manually cut off the unwanted parts of the recordings and leave only relevant piano bits.

The resulting dataset used for my experiments consisted of 16 kHz-sampled 2300 2 s piano sounds, i.e. 1.3 hours.

5.4 YouTube-8M

YouTube-8M dataset [2] is the largest multi-label video classification dataset. It is composed of 8 million videos, which amounts to approximately 500 K hours of video, annotated with a vocabulary of 4800 visual entities. The raw dataset is very robust, being several terabytes of size. For my purposes I extracted only audio from videos with the label 'piano'. Since the data comes from user-uploaded videos, the recordings are often of a lower quality, with the maximum sampling rate of 8 kHz. Due to the need of manually going through the dataset and skipping low-quality recordings, I used only a small portion of the dataset. The whole subset used for my experiments amounts to 261 source recordings, 8 s each, so approximately 35 minutes.

[3] http://magnatune.com/
[4] http://mirg.city.ac.uk/codeapps/the-magnatagatune-dataset

Chapter 6

Experimentation results

With the goal of exploring capabilities of WaveNet with a custom implementation I carried out several experiments focused on exploring different model setups, training and generation schemes. All of the generated audios and respective spectrograms, plots of waveforms, entropies, and referential recordings are available in experiments directory my github repository[1].

6.1 Model size

In order to find optimal size of WaveNet, 4 different model sizes were explored. The first one (*Medium*) is an example of an average model size that I experimented with most frequently, mainly for the reasonable trade-off between size and speed of training and generating. *Small* model is included to provide a reference point for Medium for comparison of WaveNet's performance. According to van den Oord et al., enlargement of the receptive field seems to be crucial to obtain samples that sound musical [34]. *Big* is used to explore this claim. Finally, *Big channels* increases the residual and dilation channels, as opposed to receptive field. The specifications are presented in Table 6.1.

	Sampl. rate	Dilations	Stacks	Dil. and res. channels	Skip channels
Big	8 kHz	13	5	64	1024
Big channels	8 kHz	10	5	128	1024
Medium	8 kHz	10	5	64	1024
Small	8 kHz	5	2	32	512

Table 6.1: Parameters of different model sizes.

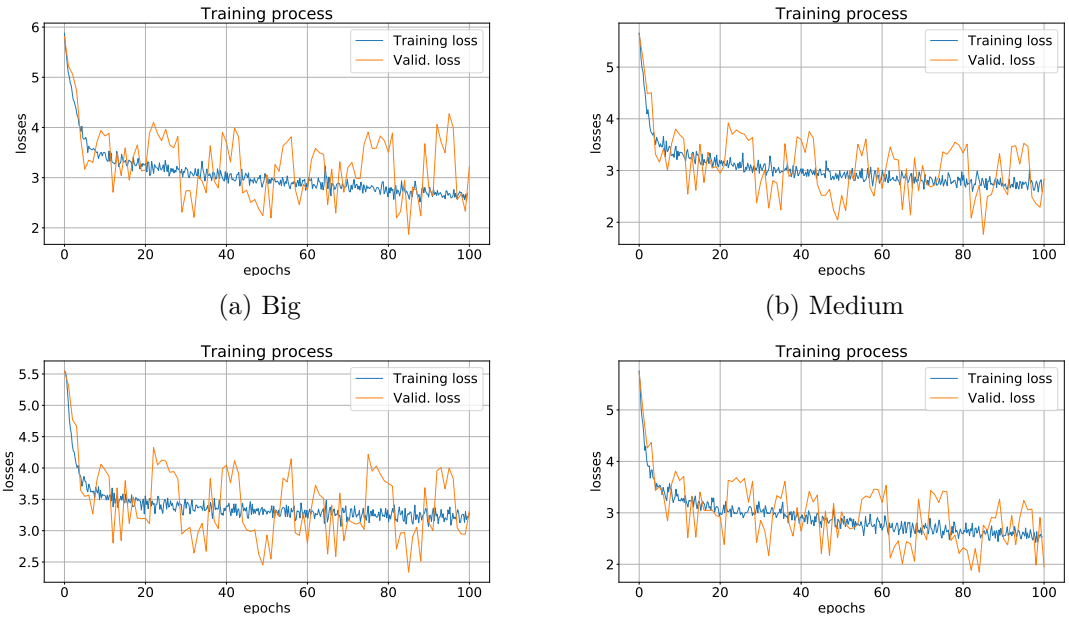
Different model sizes account for different lengths of the receptive field, as captured in Table 6.2.

[1] https://github.com/TerkaSlaninakova/BP/tree/master/experiments

	Receptive field size (n. of samples)	Receptive field size (ms)
Big	40957	5200
Big channels	5117	640
Medium	5117	640
Small	64	8

Table 6.2: Receptive field sizes corresponding to model sizes from Table 6.1, in number of samples and miliseconds, using 8 kHz sampling rate.

The models of different sizes were trained on the Youtube-8M dataset, described in Section 5.4 for 100 epochs, see Fig. 6.1.



(a) Big

(b) Medium

(c) Small

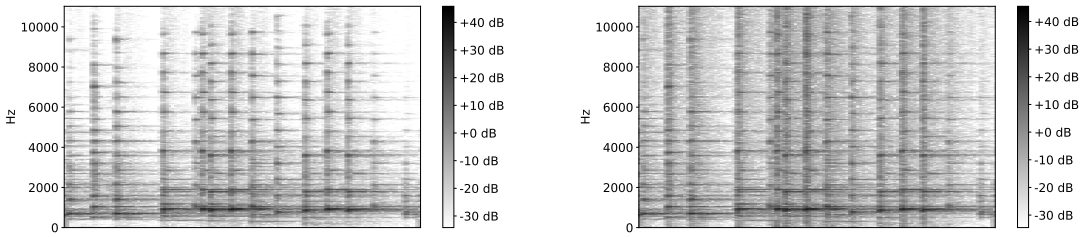
(d) Big res. and dilation channels

Figure 6.1: Training process of different WaveNet sizes. The reached training loss is about 2.75 for every model size except for small, which finished around 3.25. Validation loss is fluctuating the most for big and small model.

Fig. 6.1 suggest that medium model and model with big residual and dilation channels have slight advantage over Small and Big models in terms of the training progress.

I first explored the teacher forcing generation scheme, which produced convincing results even with the weakest (Small) model[2], see Fig. 6.2.

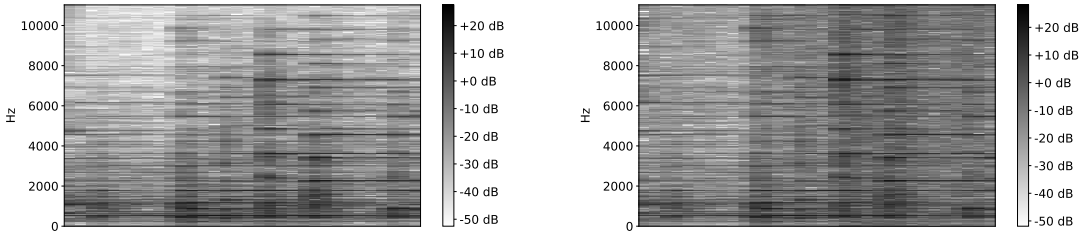
[2] https://github.com/TerkaSlaninakova/BP/tree/master/experiments/size/small/TF



(a) Referential audio (b) Generated audio

Figure 6.2: Spectrograms of referential and generated audio. There is small qualitative degradation, but the musical information was kept.

Interestingly enough, teacher forcing on a small model produces[3] convincing outcomes even when the referential audio is not from the same dataset that the model was trained on, as seen in Fig. 6.3.



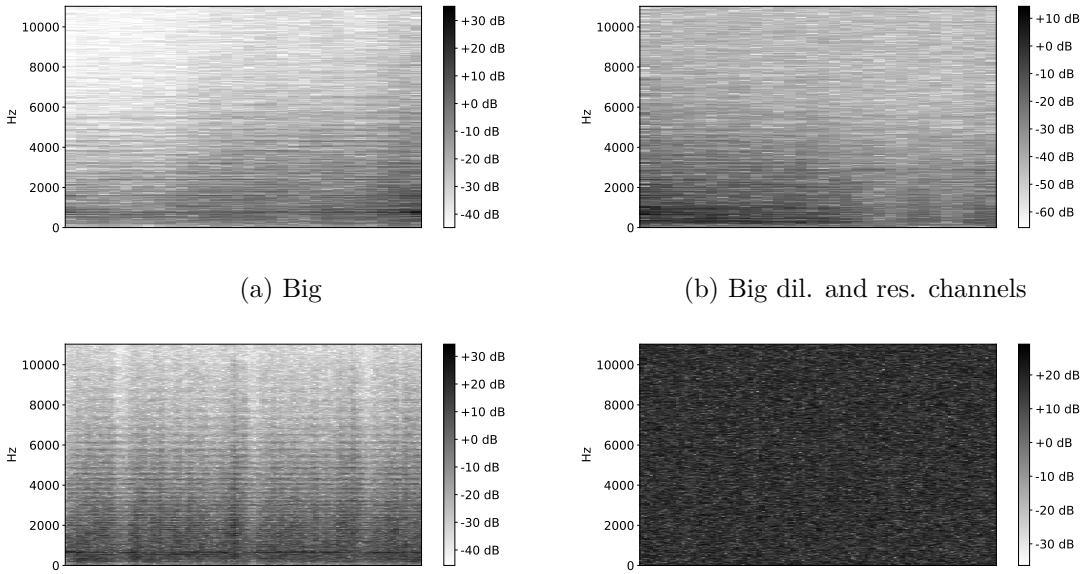
(a) Referential audio (b) Generated audio

Figure 6.3: Spectrograms of referential and generated audio, where the referential audio is not a part of WaveNet's training dataset. Produced with the Small model from 6.1.

Other models performed similarly well in the teacher forcing setup[4].

When using seeding, the differences between models became more apparent, see Fig. 6.4.

[3] https://github.com/TerkaSlaninakova/BP/tree/master/experiments/size/small/TF_notseen
[4] https://github.com/TerkaSlaninakova/BP/tree/master/experiments/size



(a) Big

(b) Big dil. and res. channels

(c) Medium

(d) Small

Figure 6.4: Models of different sizes using the seeding generation scheme. Notice the apparent aggressive noisiness of the Small model and the lack of sound of the Big model.

As for unique generation looked similar as the seeding setup in Fig. 6.4, none of the generated audios sounded musical.

6.2 Dropout

As explained in Section 2.2, dropout is used to mitigate negative effects overtraining.

When tried on medium-sized model, with the same specifications as in the previous section of the Youtube-8M piano dataset, no difference among the individual dropouts was visible, see Fig. 6.5.

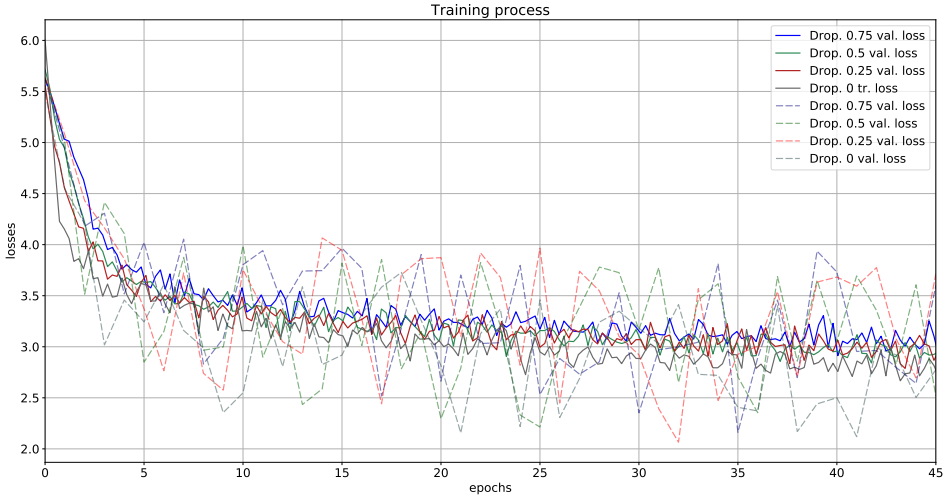
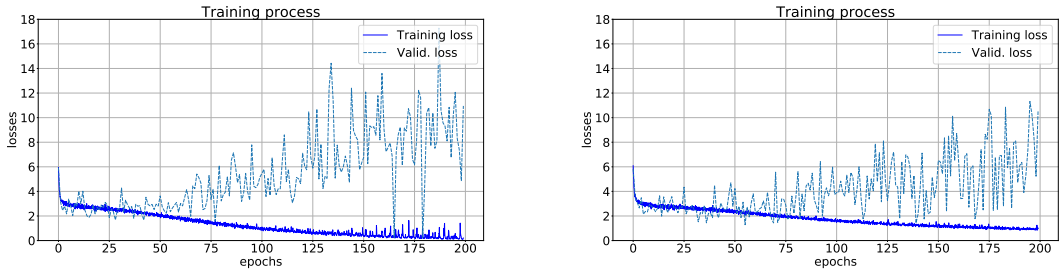


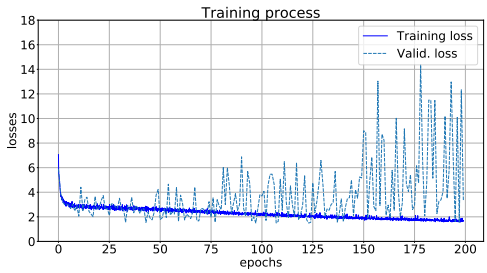
Figure 6.5: Dropout of 0, 25%, 50%, 75% applied to training of 45 epochs attempted with a medium-sized model.

To make the effect of dropout more visible, I used a variation of big model, as described in Section 6.1 with 13 dilations of 5 blocks and 128 dilation and residual channels. WaveNet was trained on the IRMAS piano dataset. Dropout ranged from 0 (no dropout, all neurons are considered in the training), 25%, 50% and 75%, meaning 75% of neurons are ignored during training. The differences in training are captured in Fig. 6.6.

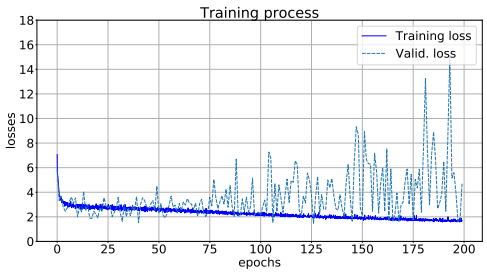


(a) Dropout 0

(b) Dropout 0.25



(c) Dropout 0.5



(d) Dropout 0.75

Figure 6.6: Training process of different dropout schemes with the same WaveNet parameters. Overtraining is the most visible when not using any dropout, as seen in Fig. 6.6a. Notice the training loss being very close to zero, while validation loss being very high. On the other hand, 75% dropout's training loss converges slowly and validation loss, although still fluctuating, is on average less than in any other dropout. The training took 26–27 hours on GeForce GTX 1080.

The resulting generated audios under seeding and unique generation schemes were not very convincing. There was no significant difference between 75% dropout[5] and no dropout[6].

6.3 Sampling rate

Audio signal with higher resolution, such as 48 kHz, captures subtle changes in the amplitude better than one with a lower resolution, e.g. 8 kHz. This is demonstrated in Fig. 5.3. In telecommunications, 8 kHz is usually deemed sufficient for speech transmission. In music production however, it is crucial to cover the whole range of human hearing (20–20,000 Hz), so following the Shannon-Nyquist sampling theorem, 44.1 kHz and higher sampling rates are often used.

It was worth trying what effect do different sampling rates have on the training process and generated audio. I chose the IRMAS' piano dataset, with 721 3 second clips originally sampled at 44 kHz and loaded it with increasing sampling rates of 8, 16, 24, 36, and 44 kHz. Figure 6.7 shows the difference between training and validation losses in regards to different sampling rates.

To preserve the same audio range (i.e. receptive field) for every sampling rate used, it was necessary to increase the size of WaveNet. The sizing adjustments are captured in Table 6.3.

[5] https://github.com/TerkaSlaninakova/BP/tree/master/experiments/drop/075
[6] https://github.com/TerkaSlaninakova/BP/tree/master/experiments/drop/0

	8 kHz	16 kHz	24 kHz	36 kHz	44 kHz
Number of blocks	5	5	5	6	7
Number of dilations	10	11	12	12	12
Receptive field (RF) size	5117	10237	20477	24572	28667
RF in Ms	630	639	853	682	651

Table 6.3: Sizing adjustments for models with different sampling rates.

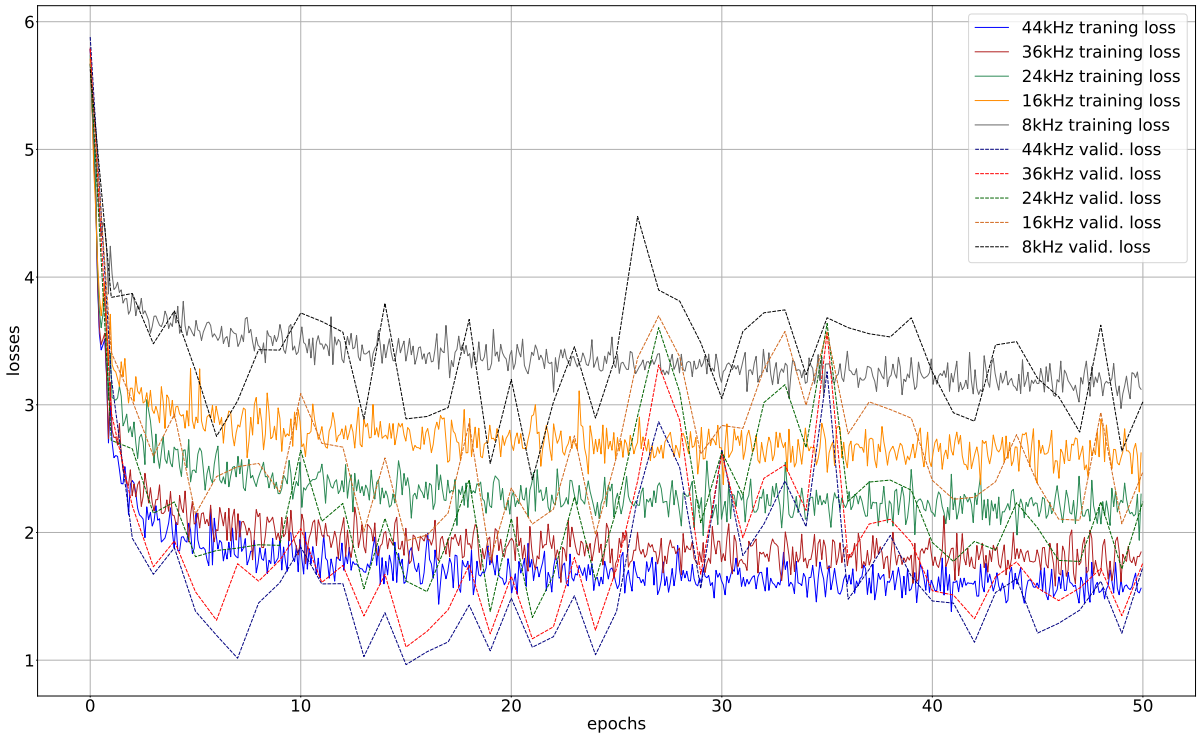
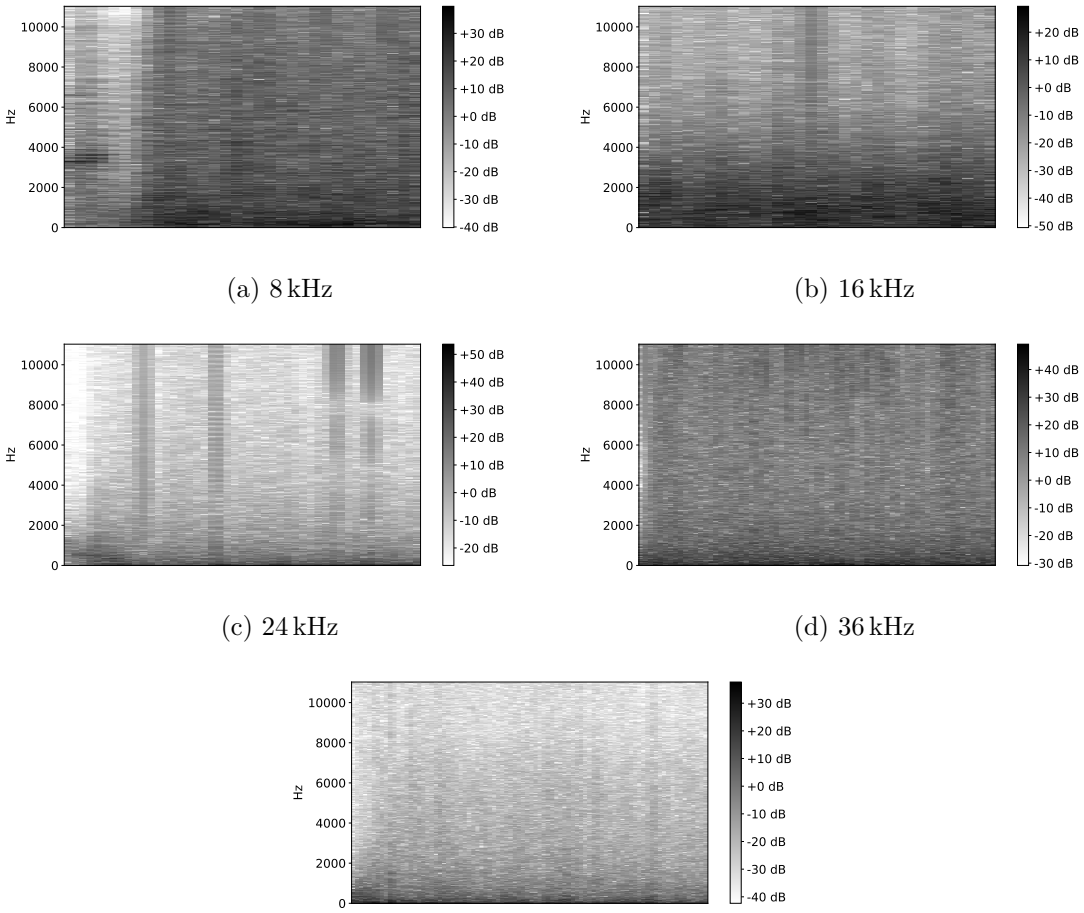


Figure 6.7: Validation and training losses of a dataset loaded at increasing sampling rates. Greater sampling rates were able to reach lower training and validation losses in a lower number of epochs, but in overall longer training time. The training times were: 8.029, 7.8, 7.36, 4.87 and 4 hours for 44 kHz down to 8 kHz respectively.

I tested the performance of the models with the seeding generation scheme[7].

[7] https://github.com/TerkaSlaninakova/BP/tree/master/experiments/SR



(a) 8 kHz

(b) 16 kHz

(c) 24 kHz

(d) 36 kHz

(e) 44 kHz

Figure 6.8: Generated audio files under the seeding generation scheme. Notice the quality is degrading after 16 kHz with higher sampling rate, which can be heard from the generated audios as well.

Since there was no benefit in continuing with higher sampling rate, as visualized in 6.8 and the training with higher sampling rates was much slower, I decided to continue with the 8 kHz model up to 500 epochs. The resulting waveforms of seeding[8] and unique data generation setup[9] can be seen in Fig. 6.9 and Fig. 6.10.

[8] https://github.com/TerkaSlaninakova/BP/tree/master/experiments/SR/8/500ep/seed
[9] https://github.com/TerkaSlaninakova/BP/tree/master/experiments/SR/8/500ep/unique

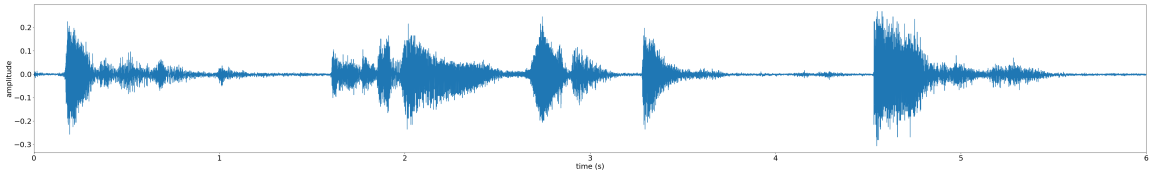


Figure 6.9: Piano under a seeding scheme after 500 epochs and with reached validation loss of 1.37 is able to produce an audio that sounds somewhat clean and musical, the individual piano keys are audible.

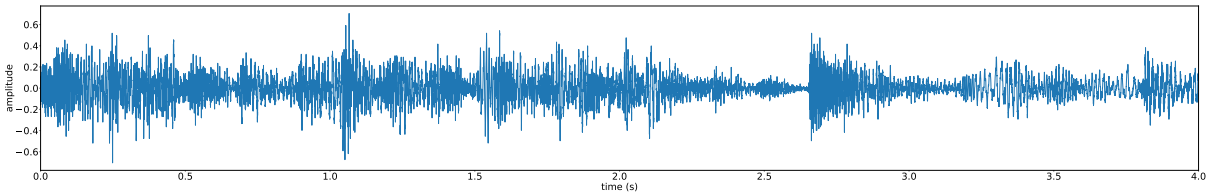


Figure 6.10: The same model as in Fig. 6.9, but producing a unique waveform.

From both waveforms in Fig. 6.9 and Fig. 6.10 some learned harmonic components are hearable. Under the unique generation scheme, the output is more noisy, as captured in the waveform plot. In the seeded generation setup, the samples corresponding to silence and shifts between notes are much cleaner.

6.3.1 Different learning rates

In scope of tuning hyperparameters, several experiments were conducted to find an optimal learning rate (LR). The effects on the training process are visualized in Fig. 6.11.

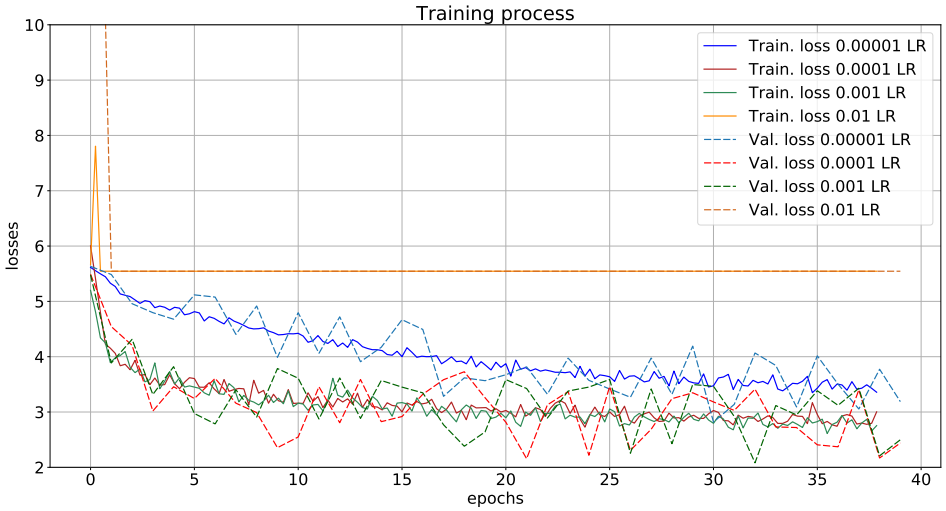


Figure 6.11: Training process of different learning rates. 10^{-2} was too high of a learning rate and caused the training process to be stuck at approximately 5.5. On the other hand, 10^{-5} was the smallest one demonstrating slower convergence, than the more optimal 10^{-3} and 10^{-4}.

As seen in Fig. 6.11, learning rates of 10^{-3} and 10^{-4} demonstrated similar training properties, while both higher and lower learning rates performed worse.

Additionally, an experiment focusing on the use of *learning rate decay* was set up. Learning rate decay refers to the gradual annealing of the learning rate over time to speed up the training convergence in the beginning and overcome overtraining later in the training process. I used tensorflow's exponential decay function[10] to implement it.

[10] https://www.tensorflow.org/api_docs/python/tf/train/exponential_decay

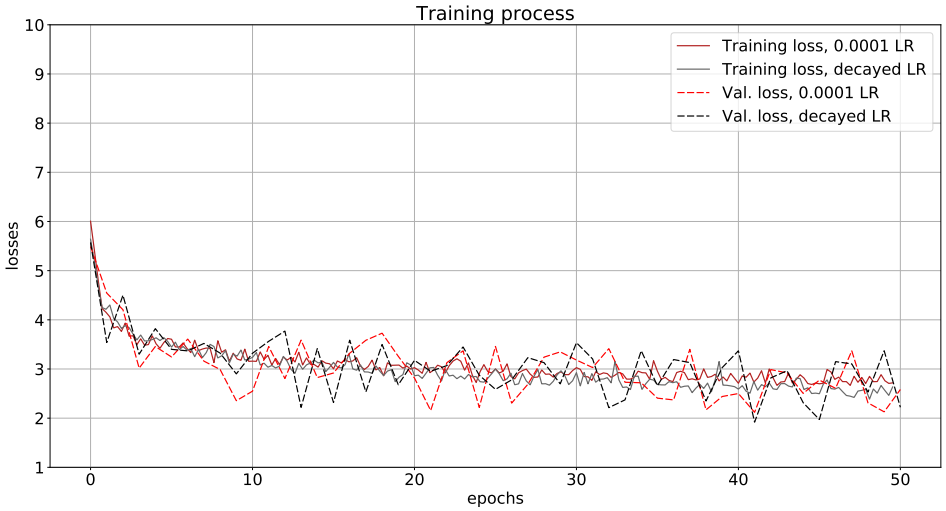


Figure 6.12: Training process of a model with standard learning rate and learning rate decay.

The initial learning rate was 10^{-3}. The final one, dropping after more than 10000 iterations, corresponding to 50 epochs as displayed in Fig. 6.12, was 0.00066. Since there was no difference between the two training processes, I decided to use the learning rate of 10^{-4} throughout the rest of the experiments.

6.3.2 Speech

Although speech was not a focus of this thesis, I carried out an experiment dedicated to it. For the sake of brevity, WaveNet was trained only a on a single speaker (p225) of the VCTK Corpus, accounting for 231 3 s recordings. The model's hyperparameters are specified in Table 6.4.

Sampling rate	Dilations	Stacks	Dil. channels	Res. channels	Skip channels
8 kHz	10	5	32	32	1024

Table 6.4: Specifications of model used for speech experiment.

The training took 250 epochs, reaching the validation loss of 0.5. Generation fared well in the teacher forcing setup[11] as expected, even with a referential recording of a different voice[12]. Unique generation produced somewhat interesting made-up language-like sounds[13], see Fig. 6.13.

[11] https://github.com/TerkaSlaninakova/BP/tree/master/experiments/voice/tf
[12] https://github.com/TerkaSlaninakova/BP/blob/master/experiments/voice/tf_notseen
[13] https://github.com/TerkaSlaninakova/BP/tree/master/experiments/voice/unique

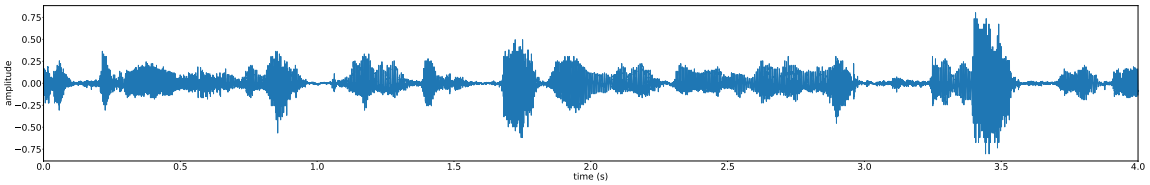


Figure 6.13: Waveform produced from unique generation scheme of WaveNet trained on speech.

6.4 Dataset-specific experiments

In these experiments, I focused on training the medium-sized model, described in Section 6.1, on the MagnaTagATune, YouTube-8M and IRMAS datasets that are described in Chapter 5.

6.4.1 MagnaTagATune

After being trained for 75 epochs, which took a little over 30 hours, I tested the model's performance on seeding[14,15].

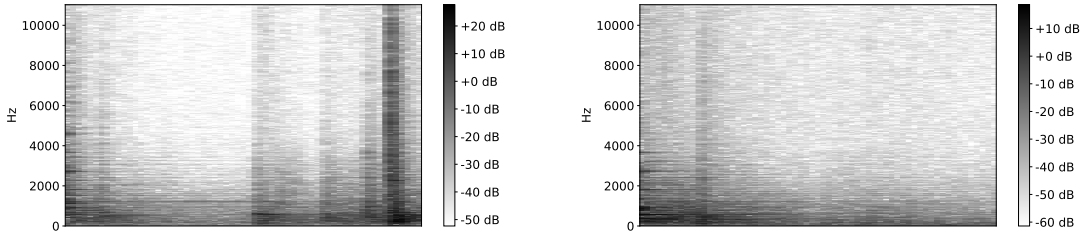


Figure 6.14: Spectrograms of the two seeded audio generations from a model trained on MagnaTagATune dataset.

The generated audios sounded rather noisy. Upon close inspection, some of the piano keys being played can be recognized, but overall the results were worse than compared to IRMAS or YouTube-8M, as confirmed by the MOS score described in 6.5.

6.4.2 YouTube-8M

After continuing with a medium-sized model trained on the YouTube-8M dataset to 200 epochs, two[16, 17] notable results were produced with the seeding setup:

[14] https://github.com/TerkaSlaninakova/BP/tree/master/experiments/other/magnatagatune/01
[15] https://github.com/TerkaSlaninakova/BP/tree/master/experiments/other/magnatagatune/02
[16] https://github.com/TerkaSlaninakova/BP/tree/master/experiments/other/yt/seed/02
[17] https://github.com/TerkaSlaninakova/BP/tree/master/experiments/other/yt/seed/01

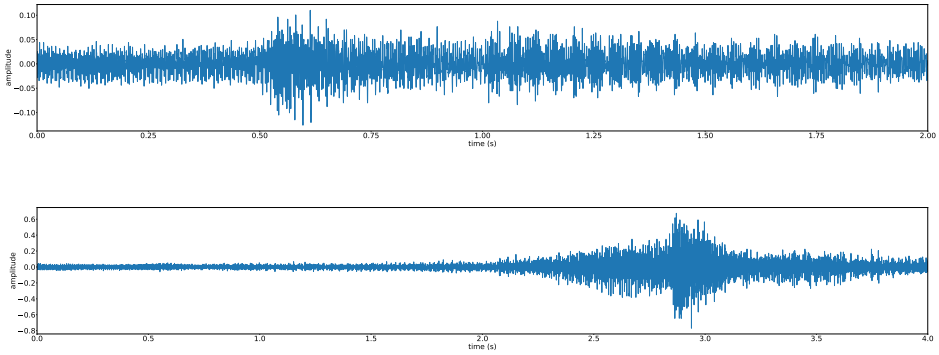


Figure 6.15: Generated audio clips after 200 epochs from the Youtube-8M dataset.

The generated audios are somewhat distant-sounding and noisy, but still with audible piano keys. The sound shown in Fig. 6.15 (upper waveform) was used in the human assessment experiments described in Section 6.5 and received a score of 3.1818.

6.4.3 Other instruments in the IRMAS datasets

IRMAS dataset contains recordings of various other instruments that were tried as well, for example flute[18], guitar[19] or violin[20].

The results from guitar generation were very noisy. This might be due to the different styles of playing, such as strumming and plucking, producing different sounds, all of which were included in the dataset.

With the seeded generation, a model trained on flute or violin was able to produce sounds with clearly audible tones corresponding to the instrument used. However, in the background, there were often distinct crashing-like noises. Unique generation was very noisy.

6.5 Human assessment

In order to meaningfully asses the quality of generated sounds, a *MOS* (*Mean Opinion score*) [21] measure was used. MOS captures subjective assessments of quality of a recording. It is expressed as a single number, typically on a 5 point scale, where 1 is lowest perceived quality, and 5 is the highest one. The overall quality is then calculated as an arithmetic mean of the individual values.

Collection of the assessments was done through a survey[21], where the subjects were asked to rate the naturalness of the listened-to recording on a five-point scale score (1: Bad, 2: Poor, 3: Fair, 4: Good, 5: Excellent). The subject had no context to the presented sounds. To provide a reference point, I included generated sounds made by the original research team as well [33]. There were 11 participants, the recordings that were used were the following:

[18] https://github.com/TerkaSlaninakova/BP/tree/master/experiments/other/flute
[19] https://github.com/TerkaSlaninakova/BP/tree/master/experiments/other/guitar
[20] https://github.com/TerkaSlaninakova/BP/tree/master/experiments/other/violin
[21] https://docs.google.com/forms/d/1Cf01MM5YElgXHSkXppVzvtAKYiwZuwztnPCRhL4sLXw

1. Piano from the experiments original research team[22]
2. Speech[23] shown in 6.13.
3. Speech from experiments of the original reseach team[24]
4. Sound generated from the IRMAS piano dataset shown in Fig. 6.9.
5. Sound generated from the YouTube-8M piano dataset, portrayed in Fig. 6.15.
6. Sound generated form the MagnaTagATune piano dataset, mentioned in 6.14

1	2	3	4	5	6
3.1818	1.81	4.545	2	3.1818	2

Table 6.5: Results of the MOS questionnaire.

From the results presented in Table 6.5, the biggest divide is between the generated speech from my experiments (1.81) and the original experiments (4.545). On the other hand, participants rated on average the audio clip form the seeded generation using Youtube-8M dataset 6.15 with the same score as the generated piano from the original experiments.

The sounds containing clear piano keys were rated lower – score of 2 for sound generated from IRMAS and MagnaTagATune datasets. The sound from YouTube-8M gained the highest score from musical audio clips. This specific sample has more of an atmospheric quality to it[25], as opposed to including sharply hearable notes, which might be why it resonated more with the subjects.

The generated speech, although seemingly promising-looking from Fig. 6.13 received the lowest score (1.81). This might be due to slight noisiness caused by unique generation scheme that was used, as opposed to a seeded one, which was employed for all the generated musical audios.

[22] https://storage.googleapis.com/deepmind-media/pixie/making-music/sample_5.wav
[23] https://raw.githubusercontent.com/TerkaSlaninakova/BP/master/experiments/MOS/2.wav
[24] https://storage.googleapis.com/deepmind-media/pixie/knowing-what-to-say/first-list/speaker-6.wav
[25] As described by one participant.

Chapter 7

Conclusion

The aim of this thesis was to explore the capabilities of a custom WaveNet implementation and compare the results with those of the original research team. Several experiments with multiple datasets were conducted to test different configurations of WaveNet to find an optimal combination of hyperparameters.

Depending on the generation setup, audios of different quality were produced. Teacher forcing proved itself to be the easiest one with output of a reasonable quality being produced just after a few hours of training using a small model. Seeding and unique generation were quite challenging, with a lot of training epochs needed to generate musically sounding outputs, as opposed to just plain noise, silence, or various non-musical noises. The results from human assessment described in Section 6.5 rated the musical samples by a score of 2–3.1818 on a 5-point scale, suggesting there is a lot of space for improvement for the sounds to be aesthetically pleasing. However, the included musical sound generated by the WaveNet's original research team received a score of 3.1818 as well, suggesting the perceived quality is comparable to the original musical experiments.

In my experiments I was able to reach the quality of WaveNet's musical clips only in one case, based on the results of human assessment. This might be due to a lack of information about the specific setup of WaveNet from the original paper, smaller datasets used, WaveNet being very resource-demanding, or limited training environment compared to the one at DeepMind used by van den Oord et al.

Finally, when experimented on speech, WaveNet produced made-up language-like sounds, suggesting human voice might be easier to generate than musical clips.

A possible extension of this work could include aiming for generation of better musical samples by constructing WaveNet to be able to handle longer training data and use bigger datasets. This would likely include further optimization of WaveNet's hyperparameters to derive the most information from the dataset during training. Another area of research might be attempting speech generation as a part of a Text-to-speech system by introducing local conditioning into WaveNet.

Bibliography

[1] Abadi, M.; Barham, P.; Chen, J.; et al.: TensorFlow: A System for Large-Scale Machine Learning. In *OSDI*, vol. 16. 2016. pp. 265–283.

[2] Abu-El-Haija, S.; Kothari, N.; Lee, J.; et al.: Youtube-8M: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*. 2016.

[3] Arik, S. O.; Chrzanowski, M.; Coates, A.; et al.: Deep Voice: Real-time neural text-to-speech. *arXiv preprint arXiv:1702.07825*. 2017.

[4] Ba, J.; Caruana, R.: Do deep nets really need to be deep? In *Advances in neural information processing systems*. 2014. pp. 2654–2662.

[5] Bengio, Y.: Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*. Springer. 2012. pp. 437–478.

[6] Blood, A. J.; Zatorre, R. J.: Intensely pleasurable responses to music correlate with activity in brain regions implicated in reward and emotion. *Proceedings of the National Academy of Sciences*. vol. 98, no. 20. 2001: pp. 11818–11823.

[7] Bosch, J. J.; Janer, J.; Fuhrmann, F.; et al.: A Comparison of Sound Segregation Techniques for Predominant Instrument Recognition in Musical Audio Signals. In *ISMIR*. 2012. pp. 559–564.

[8] Bouchard, G.; Triggs, B.: The tradeoff between generative and discriminative classifiers. In *16th IASC International Symposium on Computational Statistics (COMPSTAT'04)*. 2004. pp. 721–728.

[9] Caruana, R.; Lawrence, S.; Giles, C. L.: Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems*. 2001. pp. 402–408.

[10] Cho, Y.; Saul, L. K.: Large-margin classification in infinite neural networks. *Neural computation*. vol. 22, no. 10. 2010: pp. 2678–2697.

[11] Cisco: *Waveform Coding Techniques*. [Online; visited 17.1.2018].
Retrieved from: https://www.cisco.com/c/en/us/support/docs/voice/h323/8123-waveform-coding.html

[12] Colombo, F.; Muscinelli, S.; Seeholzer, A.; et al.: *Algorithmic Composition of Melodies with Deep Recurrent Neural Networks*. 06 2016.

[13] Delalleau, O.; Bengio, Y.: Shallow vs. deep sum-product networks. In *Advances in Neural Information Processing Systems*. 2011. pp. 666–674.

[14] Diaz-Jerez, G.: Algorithmic music: using mathematical models in music composition. *The Manhattan School of Music*. 2000.

[15] Goel, K.; Vohra, R.; Sahoo, J.: Polyphonic music generation by modeling temporal dependencies using a RNN-DBN. In *International Conference on Artificial Neural Networks*. Springer. 2014. pp. 217–224.

[16] Gonzalvo, X.; Tazari, S.; Chan, C.-a.; et al.: Recent Advances in Google Real-Time HMM-Driven Unit Selection Synthesizer. In *INTERSPEECH*. 2016. pp. 2238–2242.

[17] Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; et al.: Generative adversarial nets. In *Advances in neural information processing systems*. 2014. pp. 2672–2680.

[18] He, K.; Zhang, X.; Ren, S.; et al.: Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016. pp. 770–778.

[19] He, K.; Zhang, X.; Ren, S.; et al.: Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016. pp. 770–778.

[20] Hinton, G. E.; Osindero, S.; Teh, Y.-W.: A fast learning algorithm for deep belief nets. *Neural computation*. vol. 18, no. 7. 2006: pp. 1527–1554.

[21] Hoßfeld, T.; Heegaard, P. E.; Varela, M.; et al.: QoE beyond the MOS: an in-depth look at QoE via better metrics and their relation to MOS. *Quality and User Experience*. vol. 1, no. 1. 2016: page 2.

[22] Karlik, B.; Olgac, A. V.: Performance analysis of various activation functions in generalized MLP architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*. vol. 1, no. 4. 2011: pp. 111–122.

[23] Kim, Y.: Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*. 2014.

[24] Kingma, D. P.; Welling, M.: Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*. 2013.

[25] Krizhevsky, A.; Sutskever, I.; Hinton, G. E.: Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 2012. pp. 1097–1105.

[26] Krogh, A.; Hertz, J. A.: A simple weight decay can improve generalization. In *Advances in neural information processing systems*. 1992. pp. 950–957.

[27] Lau, M. M.; Lim, K. H.: Investigation of activation functions in deep belief network. In *Control and Robotics Engineering (ICCRE), 2017 2nd International Conference on*. IEEE. 2017. pp. 201–206.

[28] Law, E.; Von Ahn, L.: Input-agreement: a new mechanism for collecting data using human computation games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2009. pp. 1197–1206.

[29] LeCun, Y.; Bottou, L.; Orr, G. B.; et al.: Efficient backprop. In *Neural networks: Tricks of the trade*. Springer. 1998. pp. 9–50.

[30] Maas, A. L.; Hannun, A. Y.; Ng, A. Y.: Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, vol. 30. 2013. page 3.

[31] McCormack, J.: Grammar based music composition. *Complex systems*. vol. 96. 1996: pp. 321–336.

[32] van den Oord, A.: *Speech and Audio in the Northeast 2017*. [Online; visited 15.3.2018].
Retrieved from: https://avdnoord.github.io/homepage/slides/SANE2017.pdf

[33] van den Oord, A.: *WaveNet: A Generative Model for Raw Audio*. [Online; visited 29.9.2017].
Retrieved from:
https://deepmind.com/blog/wavenet-generative-model-raw-audio/

[34] van den Oord, A.; Dieleman, S.; Zen, H.; et al.: WaveNet: A Generative Model for Raw Audio. *CoRR*. vol. abs/1609.03499. 2016.

[35] van den Oord, A.; Kalchbrenner, N.; Espeholt, L.; et al.: Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*. 2016. pp. 4790–4798.

[36] Oord, A. v. d.; Li, Y.; Babuschkin, I.; et al.: Parallel WaveNet: Fast High-Fidelity Speech Synthesis. *arXiv preprint arXiv:1711.10433*. 2017.

[37] Paine, T. L.; Khorrami, P.; Chang, S.; et al.: Fast wavenet generation algorithm. *arXiv preprint arXiv:1611.09482*. 2016.

[38] Piotrowski, A. P.; Napiorkowski, J. J.: A comparison of methods to avoid overfitting in neural networks training in the case of catchment runoff modelling. *Journal of Hydrology*. vol. 476. 2013: pp. 97–111.

[39] Schittenkopf, C.; Deco, G.; Brauer, W.: Two strategies to avoid overfitting in feedforward networks. *Neural networks*. vol. 10, no. 3. 1997: pp. 505–516.

[40] Sietsma, J.; Dow, R. J.: Creating artificial neural networks that generalize. *Neural networks*. vol. 4, no. 1. 1991: pp. 67–79.

[41] Srivastava, N.; Hinton, G.; Krizhevsky, A.; et al.: Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*. vol. 15, no. 1. 2014: pp. 1929–1958.

[42] Veaux, C.; Yamagishi, J.; MacDonald, K.; et al.: CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit. 2017.

[43] Wallin, N. L.; Merker, B.; Brown, S.: *The origins of music*. MIT press. 2001.

[44] Williams, R. J.; Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. *Neural computation*. vol. 1, no. 2. 1989: pp. 270–280.

[45] Xenakis, I.: *Formalized Music: Thought and Mathematics in Composition*. Pendragon Press. 1992. ISBN 1576470792.

[46] Xiong, W.; Droppo, J.; Huang, X.; et al.: Achieving human parity in conversational speech recognition. *arXiv preprint arXiv:1610.05256*. 2016.

[47] Yamagishi, J.: *CSTR VCTK Corpus English Multi-speaker Corpus for CSTR Voice Cloning Toolkit*. [Online; visited 15.11.2017].
Retrieved from:
http://homepages.inf.ed.ac.uk/jyamagis/page3/page58/page58.html

[48] Zen, H.; Agiomyrgiannakis, Y.; Egberts, N.; et al.: Fast, compact, and high quality LSTM-RNN based statistical parametric speech synthesizers for mobile devices. *arXiv preprint arXiv:1606.06061*. 2016.

Appendix A

Human assessment results

This appendix captures the individual answers to questionnaire described in Section 6.5. Each question consisted of only a recording that the participants were asked to listen to and evaluate its quality on a scale 1-5. The figures 1.1 to 1.6 contain the distribution of the individual answers, that were summarized in Table 6.5. All of the sounds are available in my GitHub repository[1].

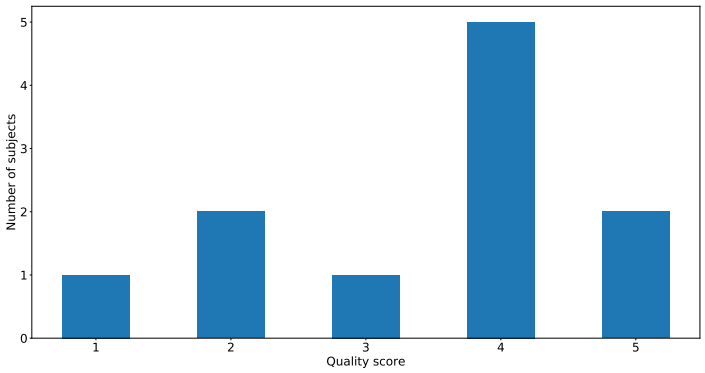


Figure 1.1: Piano from the experiments of the original research team.

[1] https://github.com/TerkaSlaninakova/BP/tree/master/experiments/MOS

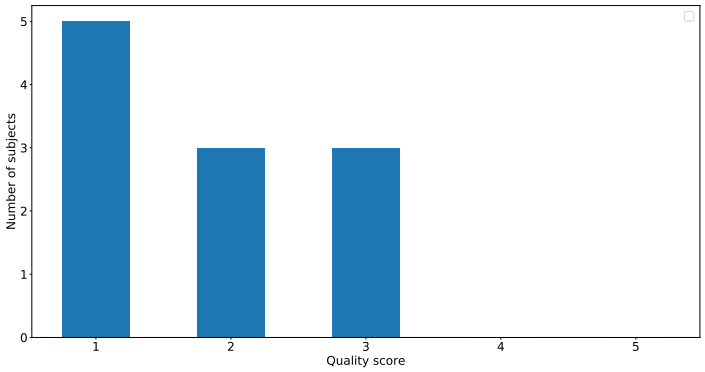


Figure 1.2: Speech from the original research team.

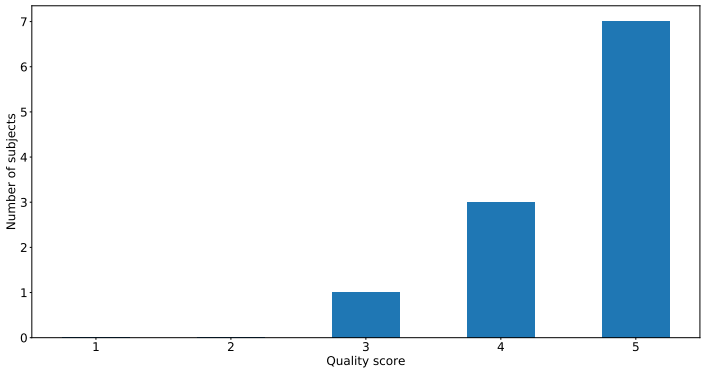


Figure 1.3: Speech from experiments of the original reseach team.

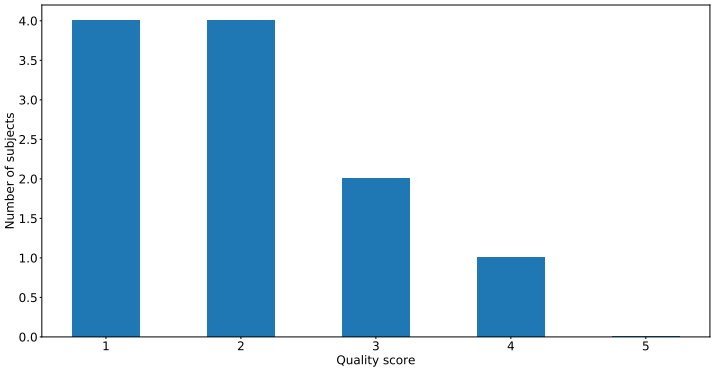


Figure 1.4: Sound generated from the IRMAS piano dataset.

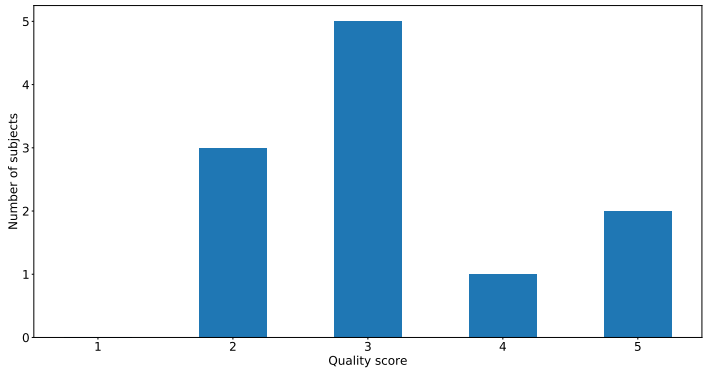


Figure 1.5: Sound generated from the YouTube-8M piano dataset.

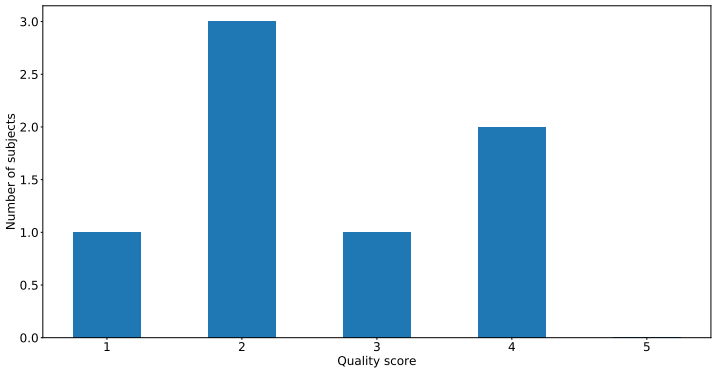


Figure 1.6: Sound generated from the MagnaTagATune piano dataset.