BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Electrical Engineering
and Communication

MASTER'S THESIS

Brno, 2021 Bc. Michele Pešán

BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF TELECOMMUNICATIONS
ÚSTAV TELEKOMUNIKACÍ

AUDIO SIGNAL MODELLING USING NEURAL NETWORKS
MODELOVÁNÍ ZVUKOVÝCH SIGNÁLŮ POMOCÍ NEURONOVÝCH SÍTÍ

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR — **Bc. Michele Pešán**
AUTOR PRÁCE

SUPERVISOR — **Ing. Štěpán Miklánek**
VEDOUCÍ PRÁCE

BRNO 2021

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

Master's Thesis

Master's study program **Audio Engineering**
specialization Zvuková produkce a nahrávání
Department of Telecommunications

Student: Bc. Michele Pešán ***ID:*** 174468

Year of study: 2 ***Academic year:*** 2020/21

TITLE OF THESIS:

Audio signal modelling using neural networks

INSTRUCTION:

Study the capabilities of neural networks based on the WaveNet architecture and networks which use recurrent layers. Focus on using such neural networks for digital black box modelling of analog effects – modulation effects, nonlinear distortion effects, etc. Summarize state of the art possibilities of using neural networks for modelling audio signals, implement neural network models in the Python programming language and use them for simulation of digital audio effects of choice.

RECOMMENDED LITERATURE:

[1] WRIGHT, Alec, Eero-Pekka DAMSKÄGG, Lauri JUVELA a Vesa VÄLIMÄKI. Real-Time Guitar Amplifier Emulation with Deep Learning. Applied Sciences [online]. 2020, 10(3) [cit. 2020-09-14]. DOI: 10.3390/app10030766. ISSN 2076-3417. Dostupné z: https://www.mdpi.com/2076-3417/10/3/766

[2] MARTÍNEZ RAMÍREZ, Marco A., Emmanouil BENETOS a Joshua D. REISS. Deep Learning for Black-Box Modeling of Audio Effects. Applied Sciences [online]. 2020, 10(2) [cit. 2020-09-14]. DOI: 10.3390/app10020638. ISSN 2076-3417. Dostupné z: https://www.mdpi.com/2076-3417/10/2/638

Date of project specification: 1.2.2021 ***Deadline for submission:*** 24.5.2021

Supervisor: Ing. Štěpán Miklánek

doc. Ing. Jiří Schimmel, Ph.D.
Chair of study program board

ABSTRACT

Through recent years, neural networks have been used more and more extensively across many science fields. Neural networks based upon the WaveNet architecture and recurrent neural networks are nowadays used in human speech synthesis and other various tasks such as *black box* modelling systems for acoustic signals alteration (modulation effects, non-linear distortion units, etc.). This academic work provides a brief introduction to the neural network terminology and common practice, elaborates on several types of neural network types with neural network audio signal modelling feasibility in mind. Furthermore describes and compares results of experimental implementation of WaveNet-style neural network and other types of neural network in black box audio signal modelling tasks.

KEYWORDS

modeling, black box, WaveNet, deep learning, neural networks, recurrent neural networks, feedforward neural networks, non-linear distortion, modulation effects

ABSTRAKT

Neuronové sítě jsou v průběhu posledních let používány stále více, a to víceméně přes celé spektrum vědních oborů. Neuronové sítě založené na architektuře WaveNet a sítě využívající rekurentních vrstev se v současné době používají v celé řadě využití, zahrnující například syntézu lidské řeči, nebo třeba při metodě *black box* modelování akustických systémů, které upravují zvukový signál (modulační efekty, nelineární zkreslovače, apod.). Tato akademická práce si dává za cíl poskytnout úvod do problematiky neuronových sítí, vysvětlit základní pojmy a mechanismy této problematiky. Popsat využití neuronových sítí v modelování akustických systémů a využít těchto poznatků k implementaci neuronových sítí za cílem modelování libovolného efektu nebo zařízení pro úpravu zvukového signálu.

KLÍČOVÁ SLOVA

modelování, black box, WaveNet, hluboké učení, neuronové sítě, rekurentní neuronové sítě, dopředné neuronové sítě, nelineární zkreslení, modulační efekty

ROZŠÍŘENÝ ABSTRAKT

Neuronové sítě jsou v průběhu posledních let používány stále více, a to víceméně přes celé spektrum vědních oborů. Neuronové sítě založené na architektuře WaveNet a sítě využívající rekurentních vrstev se v současné době používají v celé řadě využití, zahrnující například syntézu lidské řeči, nebo třeba při modelování akustických systémů, které upravují zvukový signál (modulační efekty, nelineární zkreslovače, apod.). Tato akademická práce si dává za cíl poskytnout úvod do problematiky neuronových sítí, vysvětlit základní pojmy a mechanismy této problematiky. Popsat využití neuronových sítí při modelování akustických systémů metodou *black box* a využít těchto poznatků k implementaci neuronových sítí za cílem modelování libovolného efektu nebo zařízení pro úpravu zvukového signálu.

K modelování akustických systémů byly v této práci zvoleny dvě architektury neuronových sítí. První architektura pracuje na principu vícevrstvé dopředné neuronové sítě. Druhá architektura stylu WaveNet, je dopřednou variantou neuronové sítě založené na algoritmu WaveNet. Tento druh sítí se též někdy nazývá *temporální konvoluční sítě (TCN)*. Dopředné neuronové sítě neobsahují žádné rekurentní vrstvy, spojení ani zpětné vazby. Směr prostupu neuronovou sítí je pevně daný ve směru od vstupu k výstupu sítě.

Jakožto akustické systémy k modelování byly zvoleny tři kytarové podlahové efekty typů *distortion*, *overdrive* a *delay* a jeden kytarový elektronkový zesilovač. Úkolem neuronových sítí bylo naučit se charakteristické nelinearity výše zmíněných zařízení, a tím je tedy modelovat.

Trénovací data, která neuronové sítě užívaly v průběhu trénovací fáze, byla vytvořena pomocí reamplifikace nahraných kytarových signálů výše zmíněnými efekty. Validační data pro kontrolu trénovacího výkonu sítí byla vytvořena obdobně. Vzhledem k tomu, že při reamplifikaci signálů vznikala latence zefektovaného signálu vůči nezefektovanému, bylo po skončení reamplifikace nutné provést kompenzaci dané latence. Toho se dosáhlo pomocí vzájemné korelace daných dvou signálů a následného posunutí signálu o počet rámců výsledku vzájemné korelace. Tímto způsobem se zajistila časová koherence nezefektovaného a zefektovaného signálu, která je důležitou podmínkou pro korektně nastavený učicí proces, kdy je nutné aby dva signály sobě odpovídaly v časové doméně s přesností jednoho rámce. Latence byla do signálu promítnuta v podobě reamplifikačního řetězce (DA/AD převodníky, reamplifikační box, reamplifikovaný efekt, kabeláž, popř. mikrofon). Kompenzace latence byla provedena u efektů *distortion*, *overdrive* a elektronkového zesilovače. V případě efektu *delay* by kompenzace latence naopak znemožnila modelaci daného efektu, jelikož záměrné zpoždění daného efektu by bylo kompenzováno, což by bylo pro tento druh efektu nežádoucí.

V rámci učení je neuronovým sítím předložen pár odpovídajících signálů – neze-

fektovaný a zefektovaný. Cílem sítě je poté naučit se nelineární závislosti mezi danými signály. Jedná se o metodu *učení s učitelem*. Celý dataset obsahuje zhruba 62 minut záznamu kytarového mono signálu v bitové hloubce 32 bitů a vzorkovací frekvenci 44100 Hz, ve kterém jsou zastoupeny samostatně zahrané noty i delší souvislé úseky (akordy, kousky skladeb). Zastoupeno je několik odlišných technik hry na kytaru. Celý dataset byl rozdělen v poměru 80:20 s převahou trénovací sady. Dataset byl rozdělen způsobem, který zamezoval převaze určitých druhů interpretací a technik hry v jedné ze sad. Nedodržení této podmínky by mohlo negativně ovlivnit trénovací proces.

Ke kontrole úspěšnosti trénovacího procesu slouží validační sada (kterou síť do té doby neměla možnost pozorovat), kdy již natrénované síti je na její vstup předložen pouze nezefektovaný signál a síť by již už na základě naučených zkušeností s danou nelineární závislostí měla být schopna *predikovat* zefektovaný signál. Jak úspěšná v tomto úkolu je se hodnotí porovnáním predikce sítě a zefektovaným signálem z validační sady. Tato hodnota se vypočítá pomocí kritéria ztrátové funkce. Platí, že čím nižších hodnot ztrátová funkce nabývá, tím úspěšnější síť je ve svých predikcích. V této práci byly použity dvě kritéria ztrátové funkce, a to *Mean Squared Error (MSE)* v případě vícevrstvé dopředné sítě a *Error to Signal Ratio (ESR)* v případě sítě stylu WaveNet.

Neuronové sítě byly implementovány v jazyce Python a pomocí knihovny Pytorch. Tato knihovna je specializovaná na strojové učení s využitím dedikovaných grafických karet pro akceleraci procesu. Učení neuronových sítí probíhalo nesouvisle na grafických kartách *Nvidia RTX 2080 Ti* a *Nvidia RTX 2060 Super*. V rámci úspory výpočetní kapacity při ladění trénovacího procesu byly obě neuronové sítě nejdříve natrénovány na jednom audio efektu (efektu *distortion*). Použité nastavení neuronových sítí (*hyperparametry*), které poskytovaly uspokojivé výsledky ve smyslu zvukové věrnosti zvukového modelu a výpočetní náročnosti byly poté použity i při trénování modelů zbývajících zvukových efektů. Vícevrstvá dopředná síť byla natrénována celkem se 26 sadami hyperparametrů, síť stylu WaveNet s 28 sadami. Z těch poté byla pro každou neuronovou síť a každý zvukový model vybrána konfigurace s nejnižší hodnotou ztrátové funkce. Tyto modely byly zhodnoceny ve výsledcích a predikce těchto modelů byly použity v poslechovém testu[1].

Pro rozdílnost užitých kritérií ztrátových funkcí přímé porovnání hodnot obou neuronových sítí sice nebylo možné, nicméně bylo možné kvantitativně porovnávat vlnové průběhy nebo hodnoty absolutní chyby mezi *predikovaným* signálem a původním *zefektovaným* signálem. Stejně tak bylo v rámci poslechového testu možné kvalitativně hodnotit zvukovou podobnost *predikovaného* signálu z neuronových sítí

[1]Poslechový test byl do práce zařazen nad rámec zadání jakožto doplňková metrika.

s původním *zefektovaným* signálem. K tomu byl použit test používající metodologii MUSHRA[2] test, který byl distribuován skrze webové rozhraní webMUSHRA. Vzhledem k restrikcím, které sebou přinesla pandemie Covid-19 nebylo možné uspořádat poslechový test prezenčně. Do výsledků poslechového testu se tento fakt mohl negativně promítnout v podobě zkreslení výsledků, jelikož nemohly být zaručeny konstantní podmínky pro všechny posluchače.

V rámci výsledků této diplomové práce vykazovala vícevrstvá dopředná neuronová síť lehce lepších výsledků než neuronová síť stylu WaveNet, a to především z kvantitativního hlediska, přestože predikce obou sítí byly občas navzájem poměrně blízko. Paradoxně, některé kvantitativně lépe hodnocené modely nebyly ohodnoceny jako kvalitativně lepší. Tak se například stalo v případě modelů efektu overdrive a elektronkového zesilovače. Přesto je nutné na výsledky poslechového testu nahlížet s určitým odstupem, a to kvůli výše zmíněným limitacím a s přihlédnutím k menšímu počtu respondentů (8).

Nejlepší zvukové modely obou neuronových sítí byly svými zvukovými predikcemi vzdálené state-of-the-art simulacím daných zvukových efektů. Přesto s přihlédnutím k výpočetním a časovým dispozicím se jednalo o uspokojivé výsledky odpovídající zaměření a rozsahu této práce.

[2]Vícestimulový test se skrytou referencí a kotvou

PEŠÁN, Michele. *Audio signal modelling using neural networks*. Brno: Brno University of Technology, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2021, 71 p. Master's Thesis. Advised by Ing. Štěpán Miklánek,

Author's Declaration

Author:	Bc. Michele Pešán
Author's ID:	174468
Paper type:	Master's Thesis
Academic year:	2020/21
Topic:	Audio signal modelling using neural networks

I declare that I have written this paper independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the paper and listed in the comprehensive bibliography at the end of the paper.

As the author, I furthermore declare that, with respect to the creation of this paper, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll. of the Czech Republic, Section 2, Head VI, Part 4.

Brno ....
author's signature[‡]

[‡]The author signs only in the printed version.

ACKNOWLEDGEMENT

Contents

List of Figures

List of Tables

Introduction

Digitization of music production has been a constant trend throughout the recent years and so is the demand for faithful digital emulations of analog audio effects. Many popular guitar amplifiers and distortion effects are based on analog circuitry. Such circuits utilize nonlinear components, such as vacuum tubes, diodes, or transistors to achieve the desired distortion. Digital emulations of analog systems and the adjacent field of audio signal modelling (sometimes called virtual analog modelling as well) has therefore been on rise. Once heavy, fragile and often very costly analog equipment can nowadays be, ideally, replaced by software plugins that can be run on any capable modern computer.

Several different approaches in audio signal modelling have been used over the past years. One of the most recent one, audio signal modelling using neural networks (sometimes also referred to as *deep learning* audio modelling, *end-to-end learning* for audio modelling etc.), seems especially feasible when combined with a *black box* modelling technique. This way, the modelled sound device is presented as a black box, when the inner circuitry of the given device is unknown and the only information utilized by the neural network is the input-output audio signal correlation.
As proved in [1], [2] or [3], results of this approach can be more than satisfactory, yet fairly subjective, since the quality of reproduction and "likeness" of audio device simulation or model and the reference target are a subject to a long going dispute among musicians, sounds engineers, technicians and many more.

The goal of this master's thesis is to provide an insight into the field of audio modelling, sum up the information concerning neural networks in relation to audio modelling and experimentally implement several neural network architectures that are capable of audio signal modelling. Dataset of guitar audio signals is used in the creation of a training database that can be used during the neural networks' training processes. Evaluation and comparison of the actual outcomes with the results of the latest attempts in the field of neural network audio modelling has been carried out. Listening test for evaluation of subjective quality of the audio models has been created as well.

1 Audio Signal Modelling

Audio effects modelling is the process of emulating an audio effect unit and often seeks to recreate the sound of an analog reference device. Correspondingly, an audio effect unit is an analog or digital signal processing system that transforms certain characteristics of the sound source. These transformations can be linear or nonlinear, with memory or memory-less. Most common audio effects' transformations are based on dynamics, such as compression; tone such as distortion; frequency such as equalisation (EQ) or pitch shifters; and time such as artificial reverberation or chorus [2, 4].

1.1 Nonlinear Audio Effects

These effects are widely used by musicians and sound engineers and can be classified into two main types of effects: dynamic processors such as compressors or limiters; and distortion effects such as tube amplifiers [5].

Distortion effects are mainly used for aesthetic reasons and are usually applied to electric musical instruments such as electric guitar, bass guitar, electric piano or synthesizers [5].

The main sonic characteristic of these effects is due to their non-linearity and the most common processors are overdrive, distortion pedals and tube amplifiers [5].

Dynamic range processors are nonlinear time-invariant audio effects with long temporal dependencies, and their main purpose is to alter the variation in volume of the incoming audio. This is achieved with a varying amplification gain factor, which depends on an envelope follower[1] along with a wave-shaping non-linearity. These effects tend to introduce a low amount of harmonic distortion, while for tube amplifiers a strong distortion is desired [5].

Furthermore, distortion effects and dynamic range processors are based on the alteration of the waveform which leads to various degrees of amplitude and harmonic distortion [5].

The nonlinear behaviour of certain components of the effects' circuit performs this alteration, which can be seen as a wave-shaping non-linearity applied to the amplitude of the incoming audio signal in order to add harmonic and inharmonic overtones. For example, a wave-shaping transformation depends on the amplitude of the input signal and consists in using a nonlinear function, such as an hyperbolic tangent, to distort the shape of the incoming waveform [3, 6].

[1] Envelope follower (or detector) is a utility that follows (or detects) changes in the amplitude of the input signal and recreates these changes into a control signal.

1.2 Modulation Based Audio Effects

Modulation based or time-varying audio effects involve audio processors that include a modulation signal within their analog or digital implementation [7]. These modulation signals are in the low frequency range (usually below 20 Hz). Their waveforms are based on common periodic signals such as sinusoidal, squarewave or sawtooth oscillators and are often referred to as a Low Frequency Oscillator (LFO). The LFO periodically modulates certain parameters of the audio processors, altering the timbre, frequency, loudness or spatialization characteristics of the audio. Based on how the LFO is employed and the underlying signal processing techniques used when designing the effect units, we can classify modulation based audio effects into time-varying filters such as phaser or wah-wah; delay-line based effects such as flanger or chorus; and amplitude modulation effects such as tremolo or ring modulator [3, 5].

1.3 Audio Effects Modelling

Modelling the above mentioned types of effect units or analog circuits has been heavily researched and remains an active field of research [8]. Virtual analog methods for modelling nonlinear and time-varying audio effects mainly involve circuit modelling and optimization for specific analog components such as vacuum-tubes, operational amplifiers or transistors. This often requires models that are too specific for a certain circuit or making certain assumptions when modelling specific non-linearities. Therefore such models are not easily transferable to different effects units since expert knowledge of the type of circuit being modelled is always required. Also, musicians tend to prefer analog counterparts because their digital implementations may lack the broad behaviour of the analog reference devices [3].

In terms of audio modelling methods, several different approaches are nowadays used such as *white box*, *black box* and *grey box* modelling.

White Box Modelling

White box modelling has been based on methods such as circuit simulation, where a complete study of the internal circuit is carried out. White box approach is often used in modelling of dynamic range processors, such as compressors [3].

Black Box Modelling

Black box [9] methods, such as system identification techniques, where a model is structured using only the measurements of the input and output signals. Therefore, the knowledge of the inner circuitry is omitted [3].

Grey Box Modelling

Grey box [10] technique is a combination of white and black box modelling, since some information about the circuit is known and used together with the black box approach of input-output signal relation [3].

As mentioned in [3], modelling of general purpose dynamic range compressors with black box and grey box approach has been investigated via input–output measurements and optimization routines.

1.3.1 Modelling of Nonlinear Audio Effects

Since a nonlinear system cannot be fully characterised by its impulse response, frequency response or transfer function [4], digital emulation of distortion effects have been extensively researched [8]. Different methods have been proposed such as memory-less static wave-shaping [11], where system-identification methods are used to approximate the non-linearity. Dynamic nonlinear filters, where the wave-shaping curve changes its shape as a function of the input signal or system-state variables. Circuit simulation techniques, where a complete study of the analog circuitry is performed and nonlinear filters are derived from the differential equations that describe the circuit.
Analytical methods, where the nonlinearity is modelled via Volterra series theory [12] or nonlinear black box approaches such as Wiener and Hammerstein models [3, 9, 13].

Generalization among different audio effect units is usually difficult since these methods are often either simplified or optimized to a very specific circuit. This lack of generalization is accentuated when we consider that each audio processor is also composed of components other than the non-linearity. These components also need to be modelled and often involve filtering before and after the non-linearity, as well as short and long temporal dependencies such as hysteresis or attack and release gates [3].

1.3.2 Modelling of Time-Varying Audio Effects

Most research for modelling time-varying audio effects has been explored via white box methods. In order to model the various analog components that characterize the circuitry of this type of effects, circuit simulation approaches are based on diodes, transistors, operational transconductance amplifiers (OTAs) or integrated circuits. Common methods for circuit simulation include the nodal DK-method [14] and Wave Digital Filters (WDF) [15]. By assuming linear behaviour or by omitting certain nonlinear circuit components, most of these effects can be implemented directly in

the digital domain through the use of digital filters and delay lines. Henceforth, based on all-pass filters and multiple measurements of impulse responses, a grey box modelling method for linear time-varying audio effects is proposed [3].

Recently, deep learning architectures have been explored for black box modelling of audio effects.

1.3.3 Deep Learning for Audio Effects Modelling

Deep learning architectures for audio processing tasks, such as audio effects modelling, have been investigated as end-to-end methods or as parameter estimators of audio processors. End-to-end deep learning architectures, where raw audio is both the input and the output of the system, follow black box modelling approaches where an entire problem can be taken as a single indivisible task which must be learned from input to output. The desired output is obtained by learning and processing directly from the incoming raw audio, thus reducing the amount of required prior knowledge and minimizing the engineering effort [3].

2 Neural Networks

Artificial neural networks[1] are designed in a way to mimic the style of processing of a human brain [16]. Analytically speaking of the neural network, abstraction of an optimized *nonlinear* function

$$f(x) = y \tag{2.1}$$

can be used, where the output y is an optimized function f of the input x.

In terms of hierarchy, ANN can be described as a stack of units (neurons) arranged into multiple *layers*. The key concept of neural network, an artificial *linear* neuron called Perceptron, was first introduced by Frank Rosenblatt in his work [17] in 1958. Since Perceptron's linear nature allowed only binary operations, use of an optimized non-linearity called *activation* function was proposed.

Neural network is powerful enough to solve a variety of problems that are proved to be difficult with conventional digital computational methods. The human thinking system is in parallel which means it operates with numerous of our neurons connected together. In contrast to conventional mathematical logic, the main characteristics of the human thinking process is imprecise, fuzzy, but adaptive. It learns by examples, experience and it exhibits strong adaptation to external changes. Neural networks are designed in a way to mimic most of these characteristics [16].

2.1 Artificial Neuron

Neuron is a fundamental abstract unit of a neural network. Theoretically, it can have an infinite number of inputs, but a finite number of outputs – strictly one.

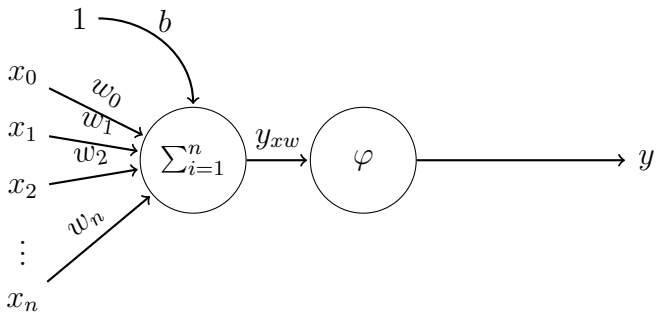


Fig. 2.1: Schematic representation of a neuron in a neural network. Neuron inputs $x_0, \ldots, x_n$ of corresponding weights $w_0, \ldots, w_n$ are being summed up. Bias b and activation function φ are being applied.

[1] Artificial Neural Network (ANN) or Neural Network (NN). These terms are interchangeable.

In an artificial neural network, weights are real numbers that tell us what importance a neuron's input has for the output. In equation:

$$y = \varphi \cdot (b + \sum_{i=1}^{n} x_i w_i)\,, \tag{2.2}$$

where inputs $x_0, \ldots, x_n$ of weights $w_0, \ldots, w_n$ create a weighted sum $\sum_{i=1}^{n} x_i w_i$ which is then multiplied by an activation function φ. This yields the true neuron's output y. Bias 1 of weight b – which can be also described as an additional neuron of input 1 – can alter the output accordingly to its weight.

2.1.1 Activation Functions

Activation (or transfer) function is used to transform the activation level of a unit (neuron) into an output signal. There are number of common activation functions in use with artificial neural networks [18], like *Sigmoid*, *Tanh*, *ReLU* or *Leaky ReLU*.

Sigmoid

Sigmoid is a nonlinear function which is described by the equation

$$\sigma(x) = \frac{1}{(1 + e^{-x})}\,. \tag{2.3}$$

It projects real valued input into a $\langle 0, 1\rangle$ range. In practice, large negative numbers are rendered almost 0 and large positive numbers are rendered almost 1. Historically, sigmoid function was heavily used because of its close resemblance to the biological neuron's firing characteristic (returns 0 or 1, "fires" or "does not fire").

Main disadvantages of sigmoid function are outputs that are not zero-centred and regional gradients that are almost zero. When the neuron's activation saturates at either tail of 0 or 1, the gradient at these regions is almost zero. If the local gradient is very small, it will influence the global gradient in a way that almost no signal will flow through the neuron to its weights and to its data as well. Extra caution is required when initialising the weights of neurons utilising sigmoid to prevent saturation. Reason behind this is if the initial weights are too large, most neurons would become saturated and the network will essentially not be able to learn [19].

Previously mentioned lack of zero-centred outputs is less severe in terms of consequences than the close to zero gradients problem. Still, it is problematic since the later layers of a neural network would receive data that is not zero-centred and therefore could introduce undesirable *zig-zagging* dynamics in the gradient updates

for the weights, since the gradient on weights w during back-propagation[2] will become either all positive or all negative [19]. Recently, the usage of sigmoid function has seen a decline since its drawbacks render it less desirable in comparison to other activation functions.

Tanh

Tanh (pictured pink in Fig. 2.2) non-linearity is similar to *sigmoid* non-linearity. Main differences are that *tanh* outputs are in the range $\langle -1, 1 \rangle$ and are zero-centred [18]. Like the sigmoid neuron, its activation saturates [19], but in practice is generally preferred over sigmoid.

ReLU

Rectified Linear Unit – ReLU (pictured magenta in Fig. 2.2), has become popular in recent years. For input $x < 0$ the output y is set to 0, but for every positive number as input it offers non-saturated linear output. Main advantage of the ReLU activation function over sigmoid or tanh is its speed. It was found to greatly accelerate the convergence of stochastic gradient descent [19, 20]. Major drawback of ReLU units is the so-called "dying ReLU" phenomenon. For example, a large gradient flowing through a ReLU neuron could cause the weights to update in such a way that the neuron will never activate on any data point again. If this happens, then the gradient flowing through the unit will forever be zero from that point on [19]. This results in a "dead" neuron[3] that cannot be used anymore in the training process. In search of a solution to this issue *Leaky ReLU* was introduced.

Leaky ReLU

Leaky ReLU (dotted orange in Fig. 2.2) is the result of one of many attempts to fix *ReLU's* main drawback. Proposed solution was to introduce a small negative slope instead of zero for inputs $x < 0$ [19].

[2] see 2.1.2

[3] Applies to a saturated sigmoid neuron as well.

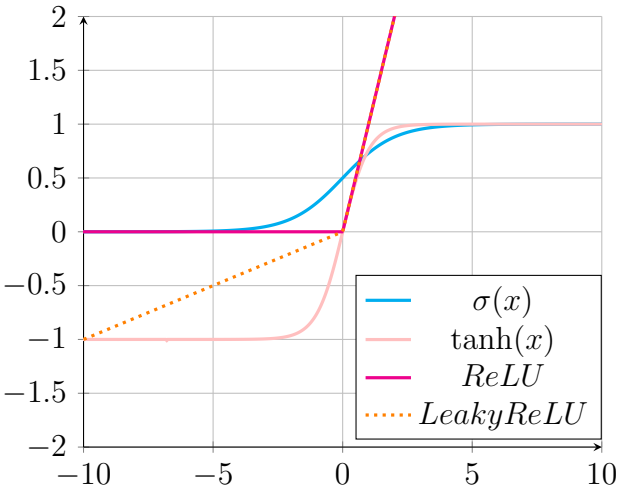
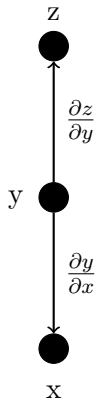


Fig. 2.2: Plotted activation functions.

2.1.2 Backpropagation

Backpropagation method is used to compute the gradient of an objective function with respect to the weights of a multi layer stack of neurons. It incorporates a practical application of the chain rule for derivatives (shown in equations 2.4). The key insight is that the derivative (or gradient) of the objective with respect to the input of a neuron can be computed by working backwards from the gradient with respect to the output of that neuron (or the input of the subsequent neuron). The backpropagation equation can be applied repeatedly to propagate gradients through all neurons, starting from the output at the top (where the network produces its prediction) all the way to the bottom (where the external input is fed). Once these gradients have been computed, it is straightforward to compute the gradients with respect to the weights of each neuron [21].



$$\Delta z = \frac{\partial z}{\partial y}\Delta y\,, \tag{2.4a}$$

$$\Delta y = \frac{\partial y}{\partial x}\Delta x\,, \tag{2.4b}$$

$$\Delta z = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\Delta x\,, \tag{2.4c}$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\,. \tag{2.4d}$$

In equations 2.4 the chain rule of derivatives tells us how two small effects (that of a small change of x on y, and that of y on z) are composed. A small change Δx in x gets transformed first into a small change Δy in y by getting multiplied by a partial derivation of y with respect to x. Similarly, the change Δy creates a change Δz in z. Substituting one equation into the other gives the chain rule of derivatives — how Δx gets turned into Δz through multiplication by the product of $\partial y/\partial x$ and $\partial z/\partial x$. Taken from [21].

2.1.3 Loss Function

Loss function, also called *cost* function or *error* function, tells us how *well* the model is performing at approximation of $y(x)$ for all training inputs x [22]. The worse the approximation, the higher the loss function.

In machine learning there are many optimization techniques such as *Stochastic Gradient Descent (SGD)*, *AdaGrad*, *RMSProp* and many combinations of these techniques such as *Adam* optimizer, which is closely described in [23].

2.1.4 Optimization

Optimization is an important part of the machine learning process. If backpropagation tells us *what* gradient our model has, the optimizer's job is to decide *how* to treat this information to minimize the *loss* function.

2.1.5 Regularization

Overfitting is one of the major issues in the machine learning and neural network field. It occurs when a model performs well on train data, but fails to generalize[4] on test data [22]. In neural network and machine learning terminology, *regularization* is a process of preventing the neural network from overfitting. Multiple regularization techniques were proposed such as *L1;L2 regularization*, *noise injection*, *error regularization*, *weight decay*, *optimized approximation algorithm*, *early stopping* [24] and *dropout* [25].

Early Stopping

Among a number of methods to avoid overfitting the early stopping using cross-validation set, the noise injection and the weight decay have been known for about two decades, however only the first one is frequently applied in practice [24]. To use an early stopping approach, apart from the training data set and the testing set, the

[4] the ability to perform well on previously unobserved data [22]

validation set is required to define stopping criteria of the learning algorithm. The ANN learning terminates when error increases for validation data, although it often continues to decrease for training data sets. When error calculated for validation data increases, while calculated for training data decreases, it is considered as fitting to the noise present in the data, instead of signal, which is a sign of overfitting [24]. Since immediate stopping upon the first error spike on validation data was not found feasible [24], a "cool-down" period called *patience* is often used. Patience stands for a number of epochs of the training process after which there is no significant change in error on validation data.

Dropout

Dropout regularization was first introduced in [25] as a technique that addresses overfitting problems in deep neural networks. The key idea is to randomly temporarily drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. During training, dropout samples from an exponential number of different "thinned" networks. At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network that has smaller weights. This significantly reduces overfitting and gives major improvements over other regularization methods [25].

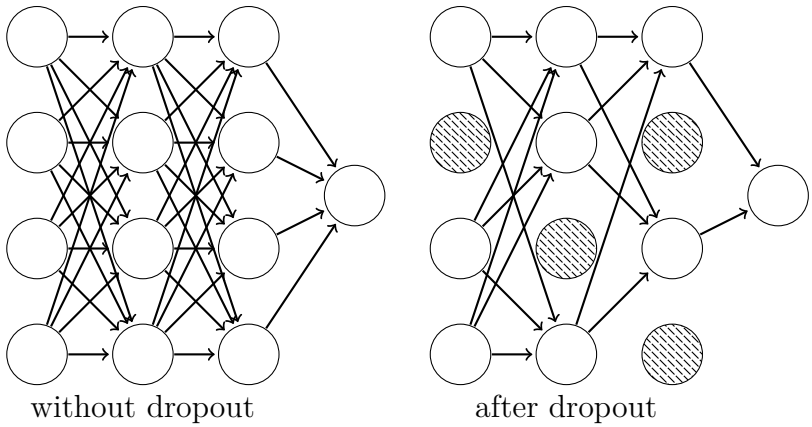


Fig. 2.3: Dropout technique depiction. Dropout has been applied to striped units in one training step.

2.2 Neural Networks Architecture

Neural networks are modelled as collections of neurons that are connected in an acyclic[5] graph. In other words, the outputs of some neurons can become inputs to other neurons. Cycles are not allowed since that would imply an infinite loop in the forward pass of a network. Neural network models are often organised into distinct *layers* of neurons.

2.2.1 Fully-connected Feedforward Neural Network

For regular neural networks, the most common layer type is the *fully-connected* layer in which neurons between two adjacent layers are fully pairwise connected, but neurons within a single layer share no connections [19].
The leftmost layer in this network is called the *input layer*, and the neurons within the layer are called *input neurons*. The rightmost or *output layer* contains the *output neurons*, or, as in this case, a single output neuron. The layer or layers in between are called *hidden layers*, since the training data does not show the desired output for each of these layers [22]. Number of layer corresponds to the term *depth*, meaning the more layers are used, the *deeper* the neural network is (eg. *Deep Neural Networks*). Networks where output from one layer is used as input to the next layer are called *feedforward* [26]. If we extend such a network to include *feedback* connections, it becomes a *recurrent* neural network [22].

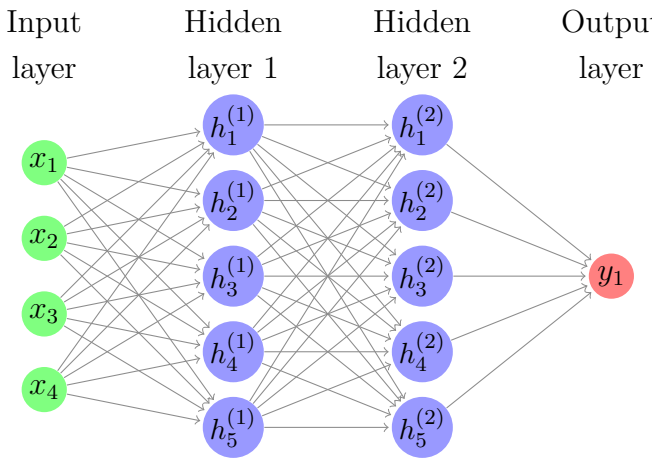


Fig. 2.4: A fully-connected feedforward neural network with two hidden layers.

[5]Exclusions apply, see 2.2.3.

2.2.2 Convolutional Networks

Convolutional networks are a specialised kind of neural network for processing data that has a known grid-like topology. Examples include time-series data, which can be thought of as a 1-D grid taking samples at regular time intervals, and image data, which can be thought of as a 2-D grid of pixels [22]. Lately, it proved itself capable in fields of speech recognition and generation(eg. WaveNet in 3). To explain the principle of operation, the input neurons are organised into a filter (eg. $N \times N$). Each neuron represents a pixel in the input data.

Convolutional Layer

From the input picture, a small region (eg. 5×5) is selected and connected to the single neuron in the first hidden layer. That region in the input image is called the *local receptive field* for the hidden neuron [26]. This region is then moved by a fixed step (*stride*) across the input picture. Every hidden neuron has its assigned local receptive field with a bias and weights. An activation function is used over every local receptive field. The same bias and weights are used for all hidden neurons in the same layer. This means that all the neurons in the first hidden layer detect exactly the same feature, just at different locations in the input image. The map from the input layer to the hidden layer is called a *feature map* [26]. Different feature maps are often stacked in the convolution layer to detect different features. The bigger the stack, the "deeper" the layer (term *deep* in this case has no connection with the number of hidden layers in a feedforward neural network).

Pooling Layer

Convolutional layer is very often followed by a *pooling* layer, although there are architectures that choose to omit the use of them. Essentially, the pooling layer simplifies the information in the output from the convolutional layer [26]. It works in a similar manner to the convolutional layer. Every unit in the pooling layer is assigned to a small section of units from the previous layer. In detail, a pooling layer takes each feature map output from the convolutional layer and prepares a condensed feature map [26]. As there are usually multiple feature maps, pooling is applied to every single one.

We can think of pooling as a way for the network to ask whether a given feature is found anywhere in a region of the image. It then throws away the exact positional information. The intuition is that once a feature has been found, its exact location is not as important as its rough location relative to other features. A big benefit is that there are many fewer pooled features, and so this helps reduce the number of

parameters needed in later layers [26]. This process is also known as dimensionality reduction.

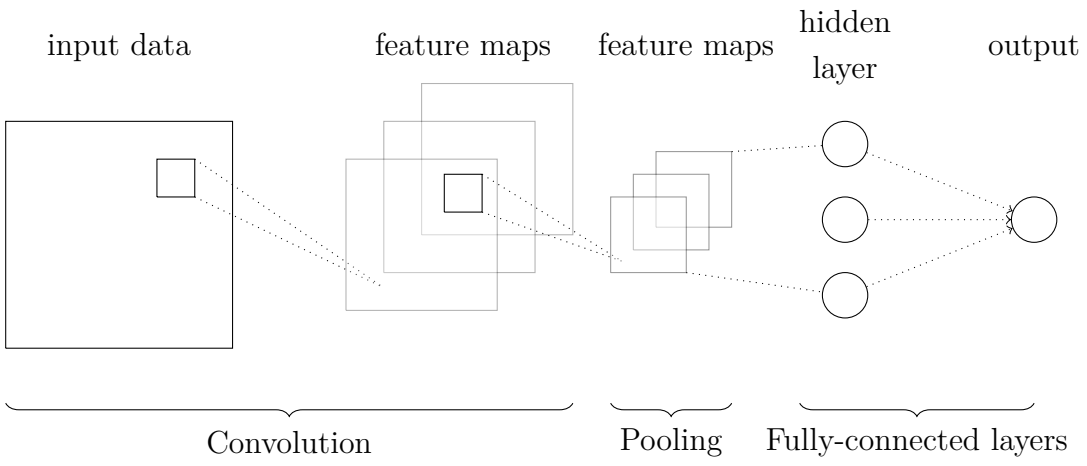


Fig. 2.5: Convolutional neural network with three convolution layers, three pooling layers and one fully-connected layer.

2.2.3 Recurrent Neural Networks

Much as a convolutional network is a neural network that is specialised for processing a grid of values such as an image, a recurrent neural network is a neural network that is specialised for processing a sequence of values. Just as convolutional networks can readily scale to images with large width and height, and some convolutional networks can process images of variable size, recurrent networks can scale to much longer sequences than would be practical for networks without sequence-based specialisation. Most recurrent networks can also process sequences of variable length [22].

Recurrent neural networks process an input sequence one sample at a time, maintaining in their hidden units a *state vector* that implicitly contains information about the history of all the past elements of the sequence. RNNs are powerful dynamic systems (yet often slow) and training them has proved to be problematic because the back-propagated gradients either grow or shrink at each time step, so over many time steps they typically *explode* or *vanish* [21].

Even if we assume that the parameters are such that the recurrent network is stable (can store memories, with gradients not exploding), the difficulty with long-term dependencies arises from the exponentially smaller weights given to long-term interactions compared to short-term ones [21].

In search of solution of the long term dependencies in RNNs, several approaches were proposed, such as use of *leaky units*, *long short-term memory* neural networks,

gated recurrent neural networks, *attention* networks or *transformers*.

Leaky Units

Leaky unit is a type of a hidden unit that has a linear self-connection with weight near one on this connection. The use of a linear self-connection with a weight near one is a different way of ensuring that the unit can access values from the past [22]. Leaky units allow the neural network to remember the input data over a long duration.

Long Short-Term Memory Network

Long Short-Term Memory (LSTM) [27] recurrent networks use *LSTM cells*, visualized in Fig. 2.6, that have an internal recurrence (a self-loop), in addition to the outer recurrence of the RNN.

Memory cell is a special unit that acts like an accumulator or a gated leaky unit [21] (eg. it is able to *remember* and intentionally *forget* data, when it is not useful anymore). It has a connection to itself at the next time step that has a weight of one, so it copies its own real-valued state and accumulates the external signal, but this self-connection is multiplicatively gated by another unit that learns to decide when to clear the content of the memory [21]. Each cell has the same inputs and outputs as an ordinary recurrent network, but also has more parameters and a system of gating[6] units that controls the flow of information [22]. LSTM networks have subsequently proved to be more effective than conventional RNNs, especially when they have several layers for each time step [21].

[6] gated unit can be controlled by another unit

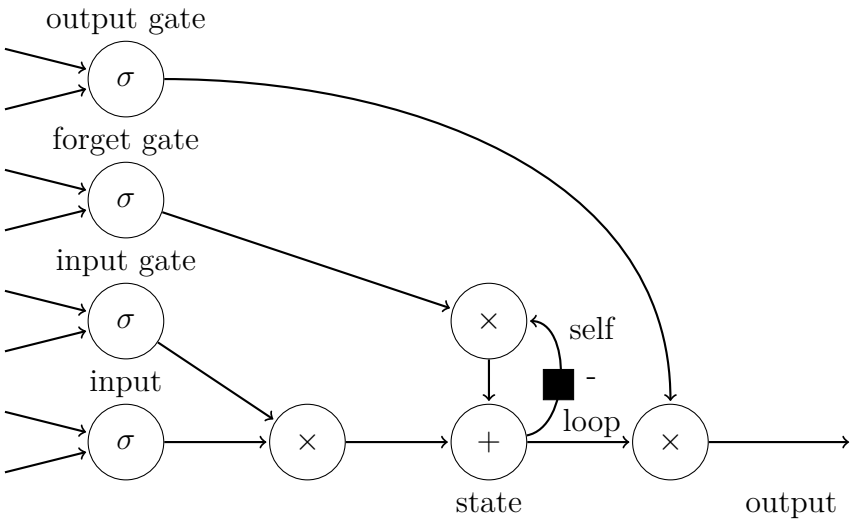


Fig. 2.6: LSTM neural network cell block diagram. Cells are recurrently connected to each other. A regular artificial neuron unit is used for computation of an input feature. Its value can be accumulated into the state if the sigmoidal input gate allows it [22]. The state unit has a linear self-loop whose weight is controlled by the forget gate. The output of the cell can be shut off by the output gate. All the gating units have a sigmoid nonlinearity, while the input unit can have any squashing nonlinearity. The state unit can also be used as an extra input to the gating units. The black square indicates a delay of a single time step [22].

Gated Recurrent Neural Network

Leaky units [28] allow the network to accumulate information (such as evidence for a particular feature or category) over a long duration. Once that information has been used, however, it might be useful for the neural network to forget the old state. For example, if a sequence is made of subsequences and we want a leaky unit to accumulate evidence inside each sub-subsequence, we need a mechanism to forget the old state by setting it to zero. Instead of manually deciding when to clear the state, we want the neural network to learn to decide when to do it [22]. This method is called a gated recurrent neural network[7].

2.3 Neural Network Training

Machine learning algorithms can be broadly categorized as *unsupervised*, *semi-supervised* or *supervised* by what kind of data is available to the network during

[7]often referred to as GRU network

the learning[8] process [22]. Supervised learning utilizes labeled data, unsupervised learning utilizes data without labels, semi-supervised learning can utilize both. The learning process is carried out on a collection of data called *dataset*. Dataset is often divided into *training* data and *test*[9]data. Training data is used for the network to train on. Test data is not accessible to the model during the training. To find out how good the model is performing, test data is utilized. Since test data is one of the major pointers telling us the model succession rate, it is necessary that there is no cross contamination (leak) between training data and test data. Semi-supervised learning is a combination of both methods mentioned above.

In machine learning there are several approaches in data modelling. These models can be roughly divided into using *generative* and *discriminative* approaches. Generative approach learns the joint probability model, $p(x, y)$, of input x and class label y, and make their predictions to compute $p(y|x)$, and then taking the most probable label y [29].

Discriminative approach models posterior class probabilities $p(y|x)$ for all classes directly and learn mapping from x to y [29].

2.3.1 Supervised Learning

Supervised learning algorithms experience a dataset containing features, but each example is also associated with a *label* or *target*. Supervised learning is about learning to predict y from x (usually by estimating $p(y|x)$), after the algorithm is shown several examples of x and an associated value y [22]. This form of learning is used furthermore in thesis.

2.3.2 Semi-supervised Learning

Semi-supervised learning is a proposed idea to find the conjunction of benefits of supervised and unsupervised learning. It uses a large amount of unlabeled data, together with labeled data, to improve the learning process. Semi-supervised learning might require less human effort in creating labeled data (features/label pairs) [30].

2.3.3 Unsupervised Learning

Unsupervised learning algorithms experience a dataset containing many features, then learn useful properties of the structure of this dataset. In the context of deep

[8]Term “training” is often used as well in the academic works or industry. They are interchangeable.

[9]There is sometimes a confusion on terminology when speaking of test data, since it is often referred to as "validation" data. In this work test and validation data are the same data.

learning, we usually want to learn the entire probability distribution that generated a dataset, whether explicitly, as in density estimation, or implicitly, for tasks like synthesis or denoising. Some other unsupervised learning algorithms perform other roles, like clustering, which consists of dividing the dataset into clusters of similar examples [22].

Essentially, unsupervised learning is a process when a neural network is shown a random sample x from the dataset and its goal is to implicitly or explicitly learn the probability distribution $p(x)$ [22]. Generative Adversarial Networks, Autoencoders or Restricted Boltzmann Machines (RBM) all utilize unsupervised learning.

3 WaveNet

Wavenet is a deep generative model of audio data that operates directly at the waveform level. It was first proposed in [31] in 2016 and is based on the PixelCNN [32] architecture. Although WaveNet's primary goal was text-to-speech modelling, when trained to model music, it was found out that it generates novel and often highly realistic musical fragments [31].

WaveNet's autoregressive model operates directly on the raw audio waveform. It uses previous time step samples to predict future samples. The joint probability of a waveform $x = \{x_1, \ldots, x_T\}$, as described in equation:

$$p(x) = \prod_{t=1}^{T} p(x_t | x_1, \ldots, x_{t-1}) \,, \tag{3.1}$$

is factored as a product of conditional probabilities as denoted in [31]. Each audio sample x_t is therefore conditioned on the samples at all previous time steps [31]. The conditional probability distribution is modelled by a stack of convolutional layers. Against the common convention in CNN, WaveNet does not utilize any pooling layers. The output of the model is a categorical distribution over the next valute x_t with a *softmax* layer and has the same time dimensionality as the input [31].

3.1 Dilated Causal Convolution

Dilated causal convolution is a combination of causal convolution and dilated convolution.

3.1.1 Causal Convolution

The key ingredient in WaveNet's operation are causal convolutions, depicted in Fig. 3.1. The use of causal convolutions ensures the proper ordering in which the model models data. The prediction $p(x_{t+1}|x_1, \ldots, x_t)$ made by the model at time step t can not depend on any of the future steps $x_{t+1}, x_{t+2}, \ldots, x_T$. At training time, the conditional predictions for all time steps can be made in parallel because all time steps of ground truth x are known. When generating with the model, the predictions are sequential. After each sample is predicted, it is fed back into the network to predict the next sample [31]. Because models with causal convolutions do not have recurrent connections, they are typically faster to train than RNNs, especially when applied to very long sequences. One of the problems of causal convolutions is that they require many layers, or large filters to increase the receptive field. As a proposed solution, the use of *dilated* convolutions was introduced. It provides an increase in

receptive field by orders of magnitude with a fraction increment of computational cost [31].

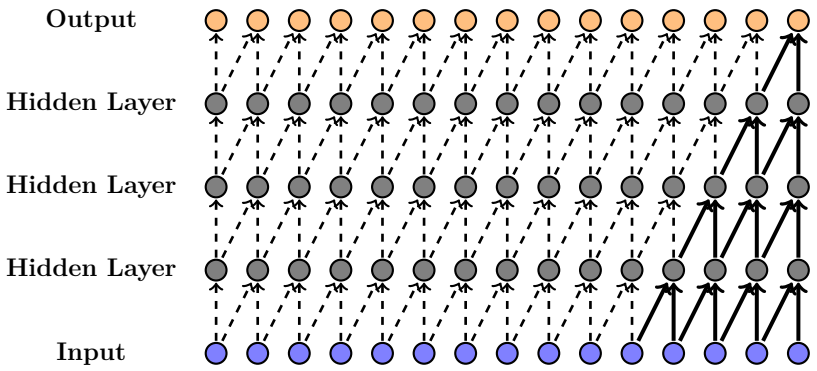


Fig. 3.1: A stack of causal convolutional layers depicted.

3.1.2 Dilated Convolution

Also called *à trous* or *convolution with holes* is a type of convolution where the filter is applied over an area larger than its length by skipping input values with a certain step. It is equivalent to a convolution with a larger filter derived from the original filter by dilating it with zeros, but it has proven to be significantly more efficient. By using dilated convolution, the network is effectively allowed to operate on a coarser scale than it would be possible with a normal convolution in use. This method is similar to pooling or convolution with stride, but in the case of dilated convolution there is no reduction in dimensionality. The output of dilated convolution has the same size as its input. In a special case when dilated convolution is utilized with dilation 1, it yields the standard convolution [31].

Layers of dilated convolutions can be arranged into *stacks*. Stacked dilated convolutions, as pictured in Fig. 3.2, enable networks to have very large receptive fields with just a few layers, while preserving the input resolution throughout the network as well as computational efficiency [31].

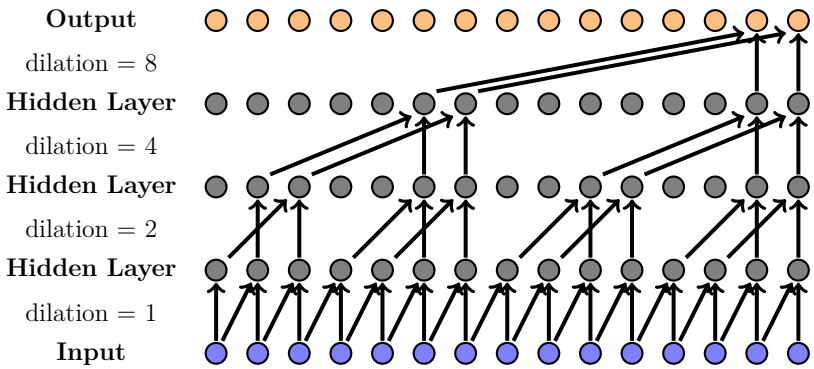


Fig. 3.2: A stack of dilated causal convolutional layers depicted.

3.2 Gated Activation Units

WaveNet utilizes the same activation function units as firstly proposed in PixelCNN architecture [32]. In equation:

$$z = \tanh(W_{f,k} * x) \odot \sigma(W_{g,k} * x)\,, \tag{3.2}$$

where W is a learnable convolution filter, k denotes the layer index, f and g stand for filter and gate, respectively. Convolution operator denoted by $*$, element-wise multiplication operator denoted $\odot$ and σ as a sigmoid activation function. This activation function was proved to be better performing for audio modelling than the ReLU activation function [31].

3.3 Residual and Skip Connections

To speed up convergence, both residual and parameterised skip-connections (Fig. 3.3) are utilized throughout the network. This also enables training of much deeper models [31]. Residual connections were proposed in [33] to solve the accuracy *degradation* issue in deep neural networks. As the depth increases, accuracy saturates and then starts to degrade rapidly. Residual connections, similarly to skip-connections, help to retain the information during the pass through the network by skipping one or several layers of the network [33].

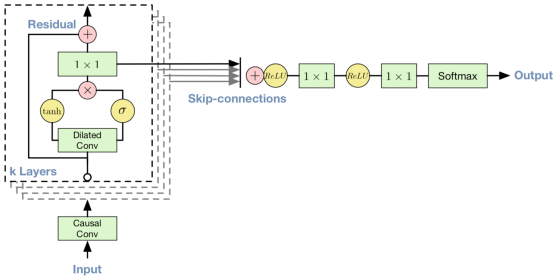


Fig. 3.3: Graphic depiction of the stacked residual blocks and the output processing cascade utilizing skip-connections. Used from [31].

3.4 Use in Audio Modelling

In [34], a feedforward, discriminative version of WaveNet architecture was proposed. This model retains WaveNet's powerful acoustic modelling capabilities, while significantly reducing its time-complexity by eliminating its autoregressive nature. Although originally proposed as a speech denoising architecture, later on, this model was adapted and used in works [35] and [1] for black box neural network audio modelling.

4 Implementation

As proposed, two types of neural networks were implemented. First type, a feed-forward type of neural network, which is an upscaled version of network that was used in the semestral thesis now serves as a baseline in comparison to the other framework – a WaveNet-style neural network. Second network, a WaveNet-style feedforward network. All frameworks were written in Python 3.6., using the Pytorch[1] 1.8. library. Experimentation was at first executed on hardware utilizing a Nvidia RTX 2080 Ti, Ryzen 3900X 12-core CPU, 32 GB of RAM and a Samsung NVMe SSD storage. Due to technical difficulties later on the experimentation had to be moved on to a different host utilizing a Nvidia RTX 2060 Super GPU, Ryzen 5 3600 6-core CPU, 48 GB of RAM and a Samsung NVMe SSD storage. TensorFlow's Tensorboard[2] package was used for the training process monitoring.

4.1 Data

Training and testing data for each sound effect consist of a set of coupled guitar audio mono signals of 32-bit float depth sampled at 44100 Hz. Fraunhofer's [36] IDMT-SMT-Guitar[3] dataset was used as a *dry* audio signal. This dataset consists of four subsets of which the second subset was used as it was deemed to be the best fit for the task. The chosen subset contains multiple playing techniques (plucking styles: finger-style, muted, picked; expression styles: normal, bending, slide, vibrato, harmonics, dead-notes). It has been recorded using three different guitars and consists of about 4700 note events with monophonic and polyphonic structure. Apart from the note events, the recorded files also contain realistic guitar licks ranging from monophonic to polyphonic instrument tracks [36]. The whole dataset (consisting of the Fraunhofer's second subset) is ~62 minutes long and it is split with an 80:20 ratio into a training and validation set. The splitting of the whole dataset was done in a manner so the recorded files would be consistently distributed across the training and validation set to avoid possible bias of spoken sets for particular types of data (e.g. playing technique).

For the creation of the *target*[4] audio signal re-amplification method was utilised as follows. Dry audio signal was sent from the DAW[5] to the reamping device[6] which ensures the line the external DA converter provides is impedance matched

[1] `https://github.com/pytorch/pytorch`
[2] `https://github.com/tensorflow/tensorboard`
[3] `https://www.idmt.fraunhofer.de/en/business_units/m2d/smt/guitar.html`
[4] term *wet* is sometimes used in this work, they are interchangeable
[5] Digital Audio Workstation
[6] Radial Engineering ProRMP Studio Reamper

and sufficiently levelled. Signal is then passed through the sound effect device and finally recorded via the mic/line input of the external AD converter[7] into the DAW. Three sound effect pedals were re-amped using this method. For re-amplification of a tube combo amplifier Sennheiser E 906 dynamic microphone was added into the signal chain.

4.1.1 Data Preprocessing

The introduced latency of the whole re-amplification chain in *target* audio signal had to be compensated so it would align properly with the *dry* audio signal. Minimal possible time shift in both signals is mandatory for the training process.

To find out whether there is a delay or *lag* between two signals, cross-correlation proves to be a robust and effective algorithm. Cross-correlation consists of the displaced dot product between two signals. It is often used to quantify the degree of similarity or interdependence between two signals[37].

Suppose that we have two real signal sequences $x(n)$ and $y(n)$. The cross-correlation sequences $r_{xy}(l)$ may be then expressed as:

$$r_{xy}(l) = \sum_{n=i}^{N-|k|-1} x(n)y(n-l)), \tag{4.1}$$

where $i = l, k = 0$, for $l \geq 0$ and $i = 0, k = l$ for $l < 0$ [37]. The algorithm shifts the $y(n)$ one sample per one step, moving it along the x-axis. According to the condition where $r_{xy}(l)$ stands for cross-correlation sequence of signals x and y, E_x and E_y are the energies of $x(n)$ and $y(n)$, respectively:

$$r_{xy}(l) \leq \sqrt{(E_x)(E_y)}, \tag{4.2}$$

which means that the auto-correlation sequence of a signal attains its maximum value at zero lag. This backs the notion that a signal matches perfectly with itself at zero shift. In the case of cross-correlation sequence, the upper bound on its values is provided in Eq. 4.2. Accordingly, when two signals align with each other at zero shift, the cross-correlation attains its maximum value.

In our case, the introduced lag between target signal $y(\mathbf{n})$ and dry signal $x(\mathbf{n})$ should be as low as 0 samples after the cross-correlation is computed and the lag is compensated, shifting the y signal accordingly to the value of lag. This is done during the preprocessing phase of data preparation by obtaining the result of both signals' cross-correlation, and compensating the lag on a sample level.

[7] Zoom TAC-2R Thunderbolt Audio Converter

4.1.2 Modelled Devices

All used audio devices feature several user adjustable controls. These controls differ in each effect, although some are common for most devices such as *Volume* or *Gain*. Individual settings of every effect are depicted in the text below.

Wem Clubman MK8 5W 1x10

As a device for the reference tube preamp tone, a Clubman MK8 amplifier combo made by Watkins Electric Music was used. It features the type ECC83 vacuum tube in the pre-amplifier stage. ECC83 tubes have been a popular choice for audio amplifiers since the introduction in 1947 [38]. In the further experiments code-named *wem*. Interface is depicted in the Fig. 4.1, used controls settings are shown in Tab. 4.1.

Fig. 4.1: Wem Clubman MK8 5W 1x10 Interface

Control	*Low*	*Mids*	*Treble*
Value	10 o'clock	12 o'clock	10 o'clock

Tab. 4.1: Wem Clubman MK8 5W 1x10 Controls Settings

Electro-Harmonix Green Russian Big Muff Pi

Firstly introduced in 1969, Electro-Harmonix Big Muff Pi has been one of the most well-known distortion pedals in the guitar and bass player's community throughout history. Even though EHX is an American company, the Green Russian (re-issued) version of the Big Muff Pi pedal is being manufactured by its sister company, Sovtek,

based in Russia. In the further experiments code-named *muff*. Interface is depicted in the Fig. 4.2, used controls settings are shown in Tab. 4.2.

Fig. 4.2: Electro-Harmonix Big Muff Pi Interface

Control	*Volume*	*Sustain*	*Tone*
Value	2 o'clock	1 o'clock	2 o'clock

Tab. 4.2: Big Muff Pi Controls Settings

Vox Ice 9 Overdrive

Vox Ice 9 Overdrive is a Joe Satriani signature overdrive pedal. It features two modes of overdrive. In the further experiments code-named *ice*. Interface is depicted in the Fig. 4.3, used controls settings are shown in Tab. 4.3.

Fig. 4.3: Vox Ice 9 Overdrive Interface

Control	*Gain*	*Tone*	*Vintage/Modern*	*Bass*	*Volume*
Value	12 o'clock	1 o'clock	Vintage	4 o'clock	12 o'clock

Tab. 4.3: Vox Ice 9 Overdrive Controls Settings

TC Electronic Flashback Triple Delay

TC Electronic Flashback Triple Delay combines three delay modules into one effect pedal. In the further experiments code-named *x4*. Interface is depicted in the Fig. 4.4, used controls settings are shown in Tab. 4.4.

Fig. 4.4: TC Electronic Flashback Triple Delay Interface

Control	*Mode*	*Time*	*Delay 1/2/3*	*Repeats*	*Mix*	*Serial/Parallel*	*Subdiv*
Value	Analog	11 o'clock	Delay 3	11 o'clock	10 o'clock	Serial	3 o'clock

Tab. 4.4: TC Electronic Flashback Triple Delay Controls Settings

4.2 Models Structure

Two types of neural networks were used for the audio signal modelling.

4.2.1 Feedforward Network

Proposed model, abstractly depicted in Fig. 4.5, is a fully-connected autoregressive feedforward neural network that utilizes *tanh* nonlinearities.

The model makes prediction $p(y_t|x_{t-1}, \ldots, x_{t-n})$ where y_t denotes an output sample at time t. x_t denotes an input sample at time t and n denotes the *context length*. Context length is the number of samples we allow the net to see before it makes a prediction. One output sample y is predicted from n input samples x. This network works natively with bit depth of 32 bits and at sampling frequency of 44.1 kHz.

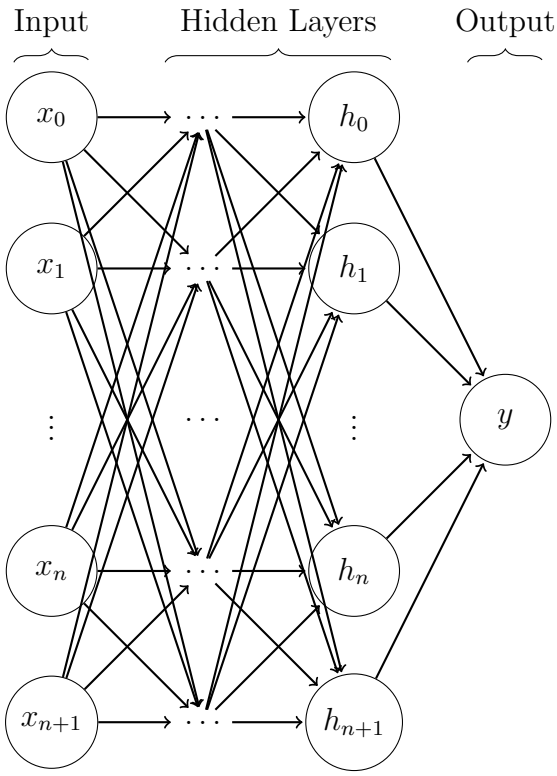


Fig. 4.5: Abstraction of the model architecture. Model consists of *tanh* activated input neurons $x_0, x_1, \ldots, x_n, x_{n+1}$, n hidden layers of *tanh* activated neurons $h_0, h_1, \ldots, h_n, h_{n+1}$ and an output neuron y with *tanh* nonlinearity activation.

4.2.2 WaveNet-style Network

WaveNet-style networks, sometimes also referred to as *Temporal Convolutional Networks* (TCN), are derived from the original WaveNet algorithm. A WaveNet-style network called PedalNetRT[8] is a neural network based on a feedforward [34] variation of the WaveNet. PedalNetRT utilizes multiple features of the original WaveNet such as dilated convolutions and gated activation units. Where it differs is audio quantization of the audio signal. The original WaveNet algorithm quantizes 16-bit audio time samples into 256 bins, and the model is trained to produce a probability distribution over these 256 possible values. In order to reduce the size of the model and increase its inference speed, the 256 channel discrete output is replaced with a single continuous output. This is done by performing a 1×1 convolution on the concatenation of each layer's output [42], as introduced in [1] and depicted in Figure 4.6.

[8] `https://github.com/GuitarML/PedalNetRT`

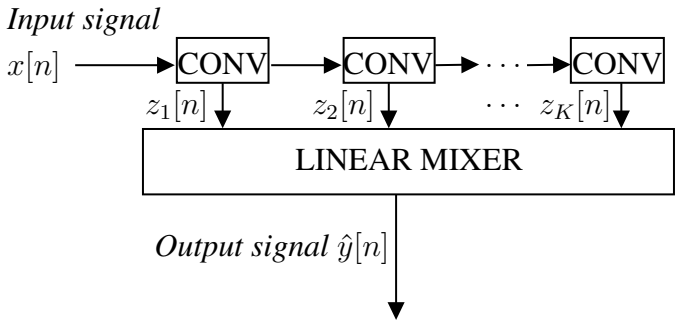


Fig. 4.6: Abstraction of the WaveNet-style neural network architecture.

The input waveform $x[n]$ is given as an input to the first convolutional layer. Inside the convolutional layer the input signal is first processed by the dilated causal FIR filter $H_k(z^{d_k})$. As k is the index of the layer, d_k states the value of the *dilation factor* of the filter. As the convolutional layers generally utilize multiple *channels*, the filtering is performed as a multiple-input and multiple-output convolution with a *kernel* H_k. This way a filter is learned for each pair of input and output channels[1].

Next the nonlinear activation function is applied to the convolution output, which yields the layer output:

$$z_k[n] = f[(H_k * x_k)[n] + b_k]\,, \tag{4.3}$$

where $f(\cdot)$ is a nonlinear activation function, $*$ denotes the convolution operator, and b_k is the learned bias term.

Convolutional layer also utilizes the residual connection that provides the input for the next layer as defined:

$$x_{k+1}[n] = W_k z_k[n] + x_k[n]\,, \tag{4.4}$$

where W_k is a 1×1 convolutional kernel controlling the summation of the convolution layer input x_k and the layer output z_k before it is given as an input to the next layer. Block diagram of a single convolution layer is shown in the Figure 4.7.

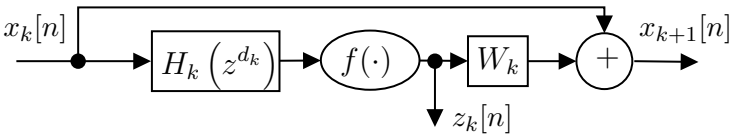


Fig. 4.7: Block diagram of a single convolutional layer.

4.3 Training

As mentioned earlier, two architectures of neural networks were implemented and put through the training process – a multilayer feedforward network and the WaveNet-style PedalNetRT network. The training process can be modified by several user adjustable arguments. These arguments are *batch size* in the case of the WaveNet-style network, and in the case of the feedforward network, *effective batch size*, *context length*, *number of neurons per layer* and *number of layers*.
Learning rate is fixed and further maintained through the training process by the Adam [23] optimizer.
Batch size parameter is the number of pairs (*dry* and *target* samples) which the network sees in each optimization step.
Gradient accumulation step parameter adjusts the optimizer's behaviour, preventing it from making an optimization step by accumulating the gradients for a given number of steps. In our case this yields more smoothed out gradients, as it mitigates the excessive *zig-zagging* of the optimizer when trying to figure out the optimal path through the neural network manifold towards the minimum.
Effective batch size parameter is the batch size times the gradient accumulation step.
Number of epochs denotes the number of complete passes through the data.
Context length parameter, first described in 4.2.1, specifies how many samples the network sees to make a prediction of one sample. It also specifies the number of neurons in the input layer of the network.
Number of layers defines the depth of the neural network.

The key to the ablation process is finding an optional state between computation difficulty and satisfactory results. Sets of hyperparameters provided below were chosen following the ablation guidelines as defined. Due to the time and computational resources restrictions, ablation was done using the Electro–Harmonix Big Muff Pi distortion pedal (later referred to as *muff*) data and these sets of hyperparameters then used in the training process of the rest of the sound effects. This approach is sometimes referred to as transfer learning.

4.3.1 Feedforward Model

The feedforward model utilizes several settings (see Tab. 4.5) of the arguments mentioned above. These sets were picked after experimentation with the goal of finding the optimal sweet spot of feasible results in respect to the computational difficulty and the sound quality. The feedforward neural network was trained for 10 epochs per each model.

Mean Squared Error

In neural network training several criterions can be used in evaluation of the nets' learning performance stating the loss value. In theory (omitting overfitting scenarios) the lower the loss value, the better learning performance. This value is computed via several formulas, all providing the insight into the learning process. As for this thesis, the *Mean Squared Error (MSE)* criterion was used for loss value computation of the feedforward neural network. As defined in:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2 \tag{4.5}$$

Where n denotes the number of samples, y_i is the target value, $\hat{y_i}$ value predicted by the network.

Effect Code-name	Best Validation Loss (MSE)	Input Context	Network Structure	Total Trained Parameters
ice	0.00011	2048	2048-1024-512-256-1	2754561
muff	0.00735	2048	2048-1024-512-256-1	2754561
wem	0.00001	2048	2048-1024-512-256-1	2754561
x4	0.00040	1024	1024-512-512-256-1	919041

Tab. 4.5: List of the best settings used in the training of the feedforward network providing the lowest validation loss. For the full list of tried out settings, see A.

Description of the header in the table 4.5 goes as:

"Network Structure" – describes the structure of the network in terms of layers of neurons (e.g. 2048-512-512-1 denotes a feedforward network with the input layer of 2048 neurons, first hidden layer of 512 neurons, second hidden layer of 512 neurons and the output layer consisting of 1 neuron).

"Total Trained Parameters" – number of parameters used by the network in the training process.

As visible from the contents of the Table 4.5, a single combination of settings showed the best results in the training of the *ice*, *muff*, and *wem* effects. The only exception was the *x4* model that favoured smaller network structure to achieve the best results.

Notable exception in the term of the loss function values was the *wem* model that reported several magnitudes smaller numbers. This might support the notion that learning of the spoken *target* signal was a fairly easy task for the feedforward network.

4.3.2 WaveNet-style Model

The excerpt of settings used for the training of the WaveNet-style PedalNetRT best performing models can be seen in the Table 4.6. These sets were picked following

the same ablation process as for the feedforward network. The WaveNet-style neural network was trained for 150 epochs per each model.

Error to Signal Ratio (ESR)

Error to Signal Ratio (ESR) loss criterion is similar to the *Mean Squared Error (MSE)*. The denominator in the *ESR* is used to normalize the loss value with regards to the energy of the target signal, preventing the loss from being over influenced by the segments of signal with higher energy [1]. Denoted as:

$$ESR = \frac{\sum_{i=1}^{n}(y_i - \widehat{y_i})^2}{\sum_{i=1}^{n} {y_i}^2} \tag{4.6}$$

Where n denotes the number of samples, y_i is the target value, $\hat{y}_i$ value predicted by the network.

Effect Code-name	Best Validation Loss (ESR)	Input Context	Network Structure	Total Trained Parameters
ice	0.16888	4410	WaveNet-12-10-1-3	10585
muff	0.81137	4410	WaveNet-12-10-1-3	10585
wem	0.30842	13230	WaveNet-12-10-1-3	10585
x4	0.72024	4410	WaveNet-12-10-1-3	10585

Tab. 4.6: List of the best settings used in the training of the WaveNet-style network providing the lowest validation loss. For the full list of tried out settings, see A.

Description of the header in the Table 4.6 goes as:

"Network Structure" – describes the inner architecture of the convolution layer. E.g. settings 12-10-1-3 denotes the network that utilizes 12 *channels* per convolution layer, *dilation factor* of 10 *repeated* once, and *kernel* of size 3. These were default settings supplied with the PedalNetRT proven to yield the best results in comparison to the computational load, as provided by [42].

"Total Trained Parameters" – number of parameters used by the network in the training process.

5 Results

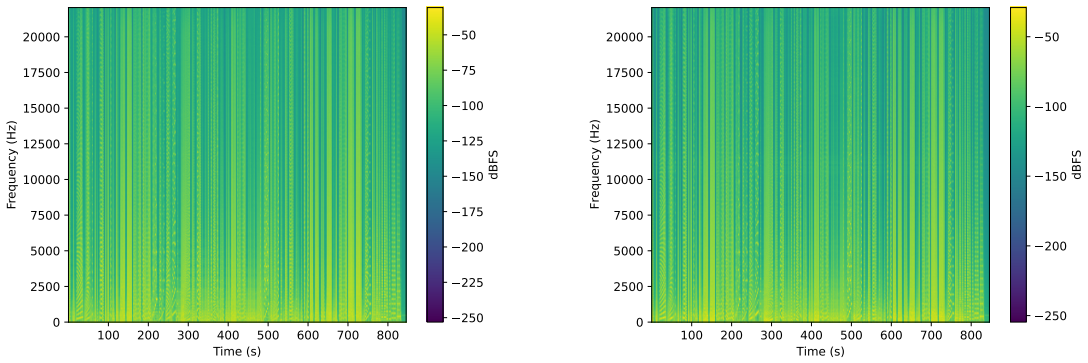
Concerning audio signal modelling, how sound is perceived by the human ear is fairly crucial in the task. Ideally, the simulation model of the audio device should be indistinguishable from the modelled audio device. How likely for an audio model this is to be achieved and whether this is even achievable is a subject of long going dispute. At this point the quantitative (or objective) and qualitative (or subjective) terms should be introduced. This chapter can be then divided into two subchapters, quantitative results and qualitative results, that are more or less interconnected. In qualitative subchapter plots are evaluated and discussed, qualitative subchapter yields results and conclusion on the listening test.

5.1 Quantitative Results

Quantitative results provided via loss functions, waveform, and spectrogram plots. It provides us an insight into how successful the network was in modeling an audio device, how similar (or different) the waveforms are to each other, etc. Predictions from the both neural networks are compared per each effect.

5.1.1 Electro-Harmonix Green Russian Big Muff Pi

The *muff* model was modelled after the only distortion effect in the pool, the Electro-Harmonix Green Russian Big Muff Pi. It has proved to be the most tricky one since the distortion is fairly extensive and both networks struggled similarly to model its features faithfully. Even though difference spectrograms in Fig. 5.1 shows roughly similar outcome, the waveform plots and absolute error comparisons of the *target* and *predicted* signals (Fig. 5.2, 5.3), and detailed plots Fig. 5.4, 5.5 respectively, provide an insight into the performance of waveform prediction of both networks, making the feedforward network model a better performing one. Still both networks provided a fair take on the heavy distortion, as is further discussed in 5.2.



(a) Difference spectrogram of the *predicted* signal made by the feedforward *muff* model and the corresponding *target* signal.

(b) Difference spectrogram of the *predicted* and *target* signals made by the WaveNet-style network *muff* model.

Fig. 5.1: Difference spectrograms of the *muff* models provided by the feedforward network (a) and the WaveNet-style network (b).

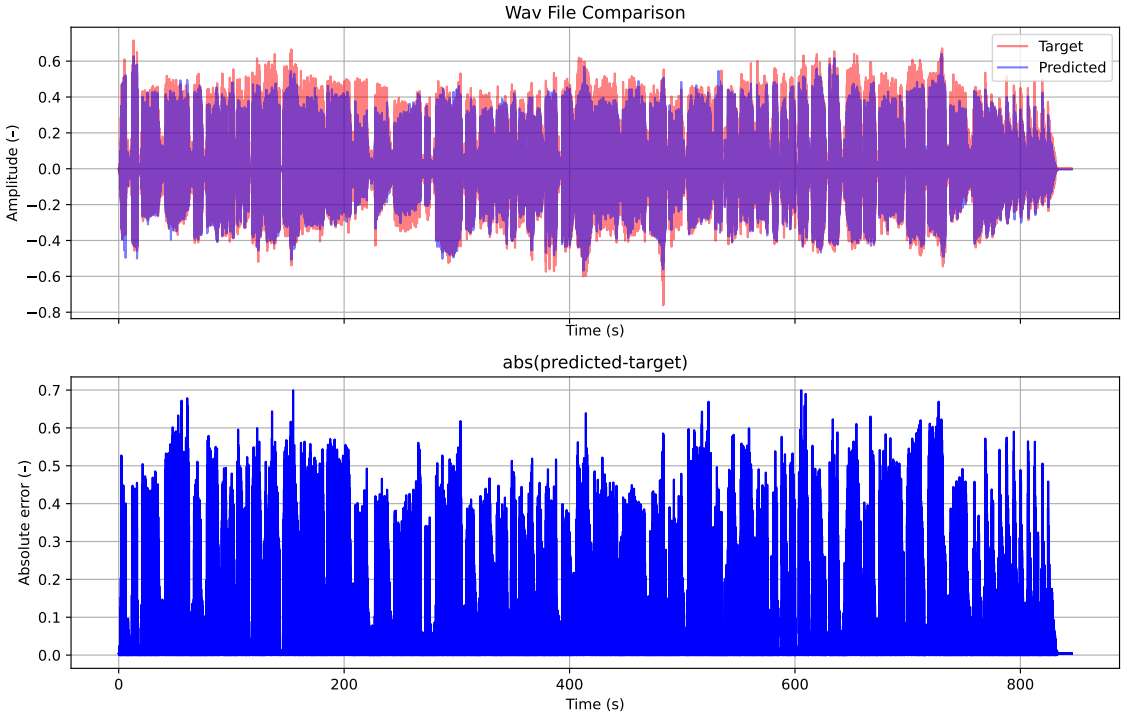


Fig. 5.2: Waveform and absolute error plots of the feedforward *muff* model.

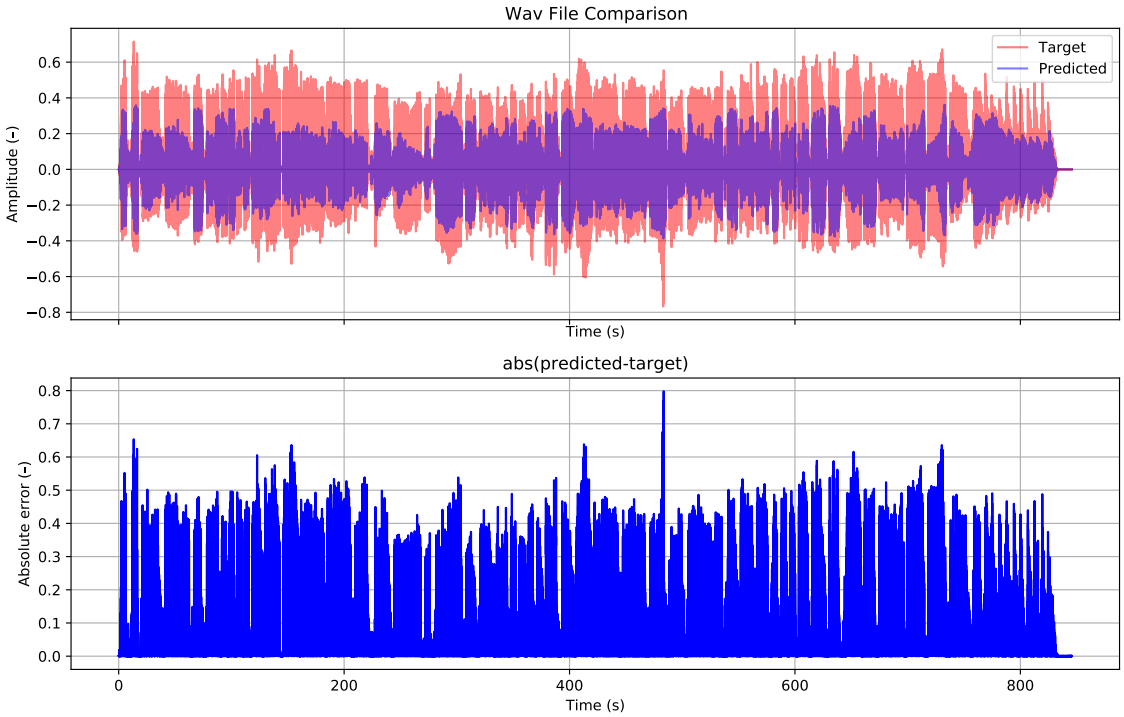


Fig. 5.3: Waveform and absolute error plots of the WaveNet-style *muff* model.

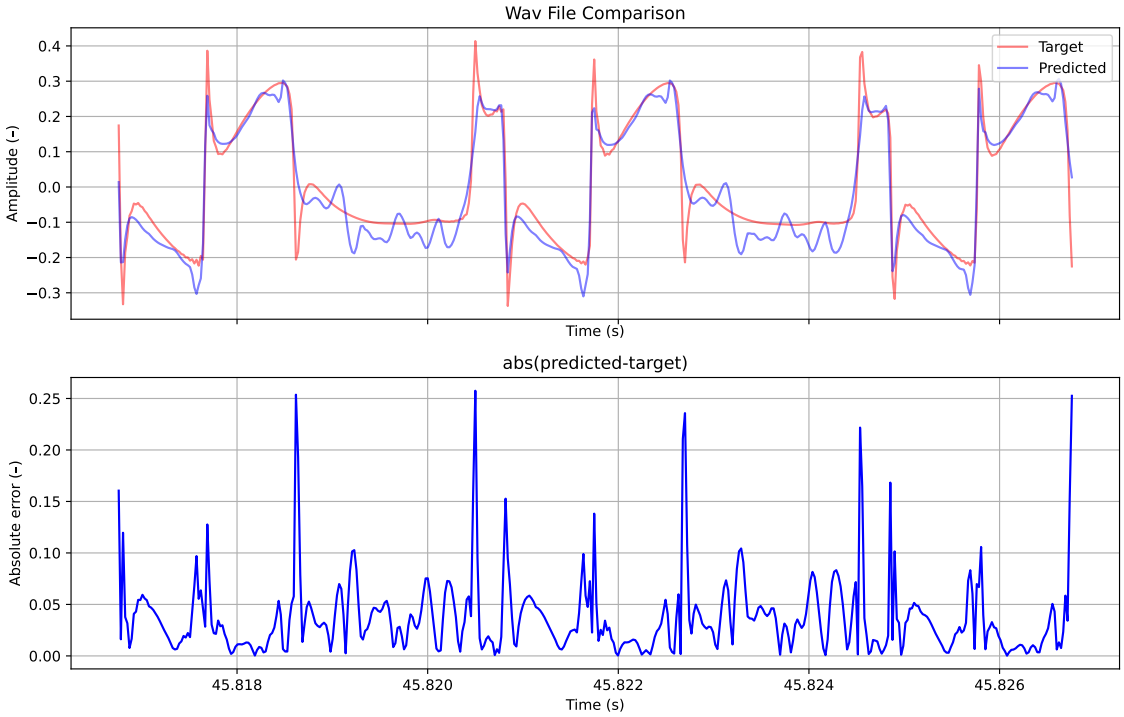


Fig. 5.4: Detailed waveform and absolute error plots of the feedforward *muff* model.

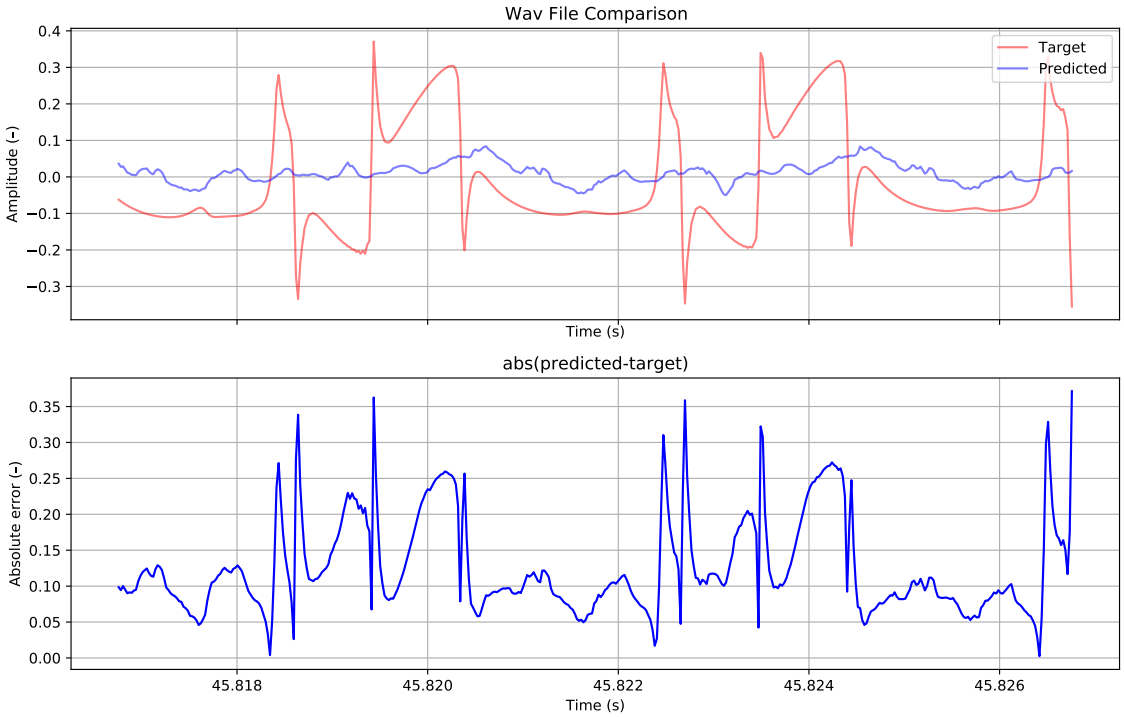
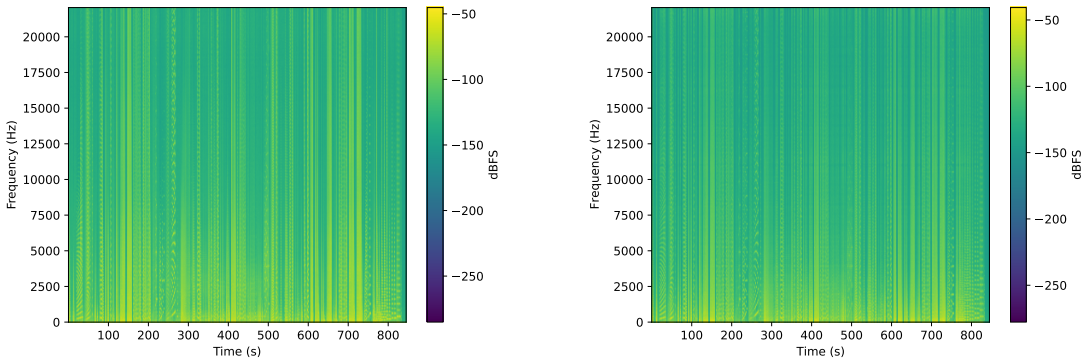


Fig. 5.5: Detailed waveform and absolute error plots of the WaveNet-style *muff* model.

5.1.2 Vox Ice 9 Overdrive

Both networks seemed more successful in modelling of the Vox Ice 9 Overdrive pedal effect. The reasoning behind this might be the nonlinearity introduced into the signal was not as major as it was in the distortion pedal. Evaluation of the difference spectrograms in Fig. 5.6 shows similar features in both spectra. Waveform comparisons of the *target* and *predicted* signals (Fig. 5.7, 5.8) and the detailed comparisons of the waveforms (Fig. 5.9, 5.10) show similar outcomes of both networks standing quite closely to each other with their predictions.



(a) Difference spectrogram of the *predicted* signal made by the feedforward *ice* model and the corresponding *target* signal.

(b) Difference spectrogram of the *predicted* signal made by the WaveNet-style *ice* model and the corresponding *target* signal.

Fig. 5.6: Difference spectrograms of the *ice* models provided by the feedforward network (a) and the WaveNet-style network (b).

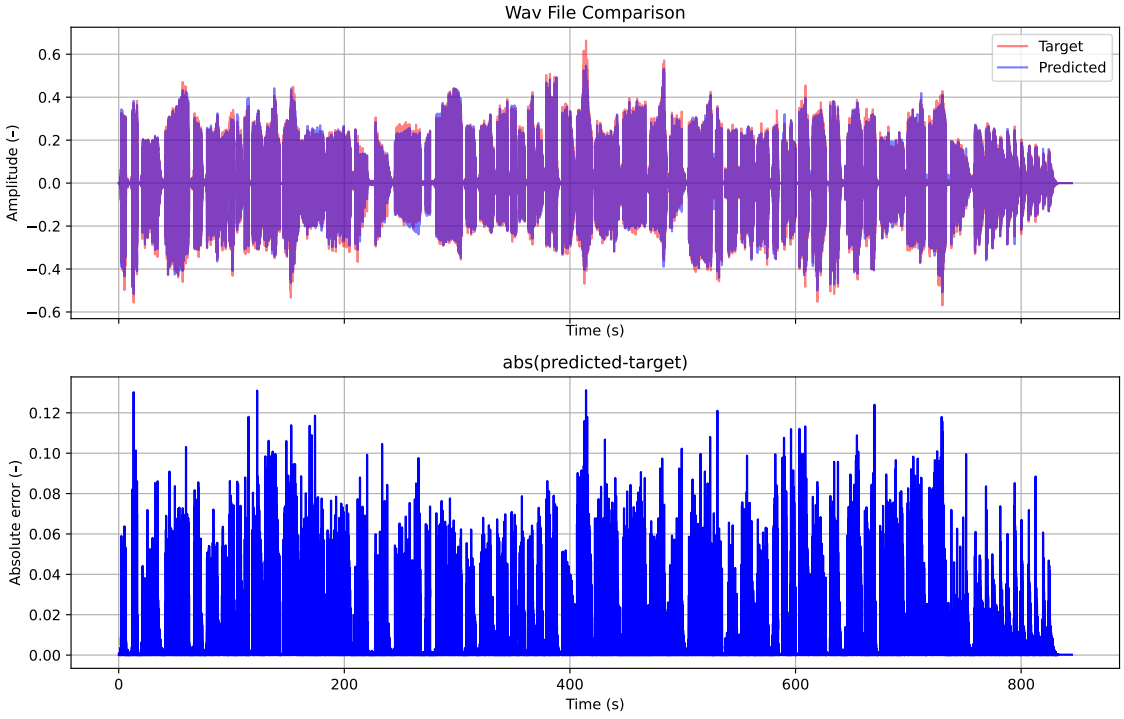


Fig. 5.7: Waveform and absolute error plots of the feedforward *ice* model.

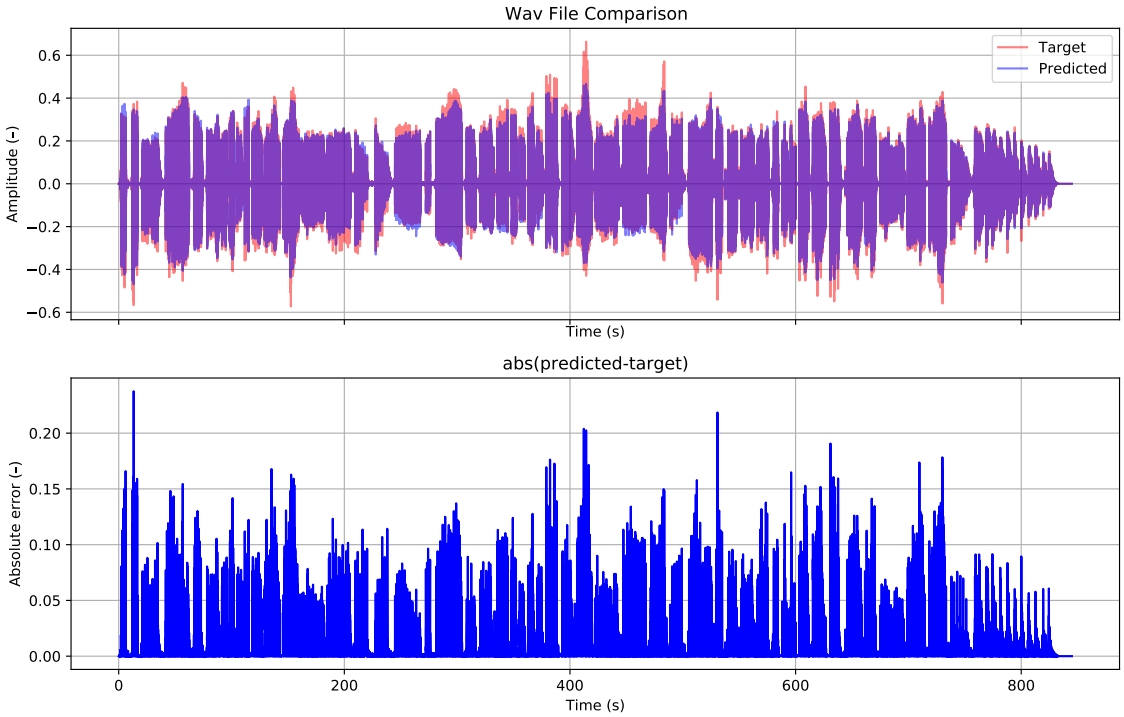


Fig. 5.8: Waveform and absolute error plots of the WaveNet-style *ice* model.

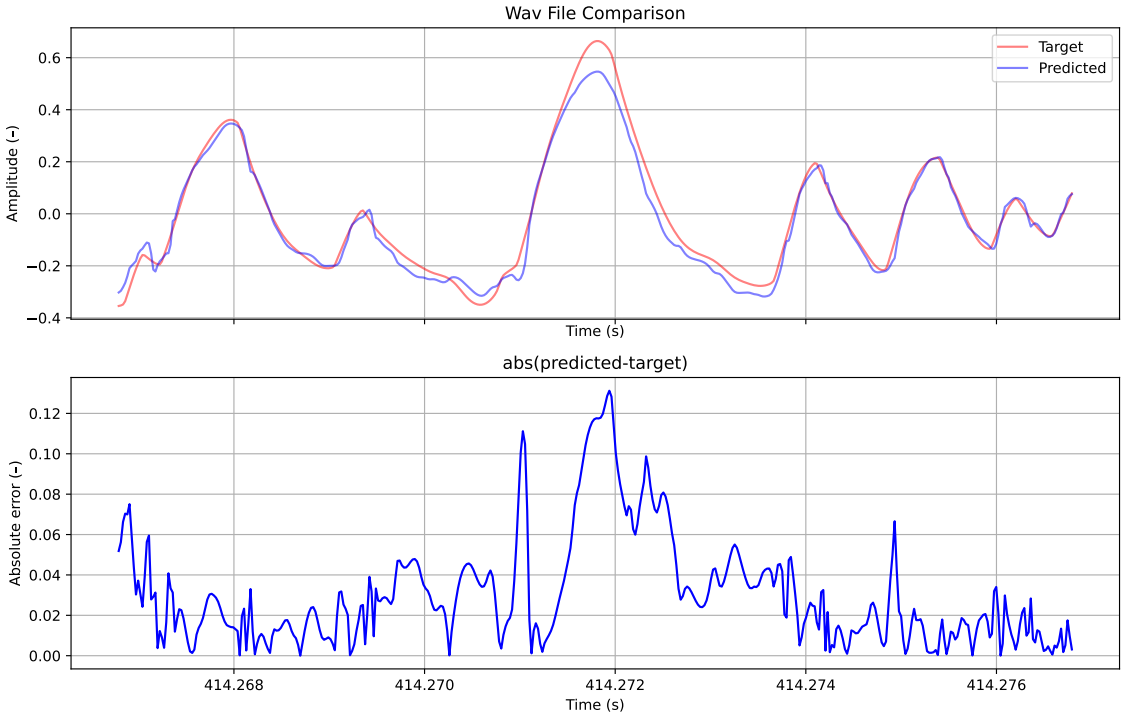


Fig. 5.9: Detailed waveform and absolute error plots of the feedforward *ice* model.

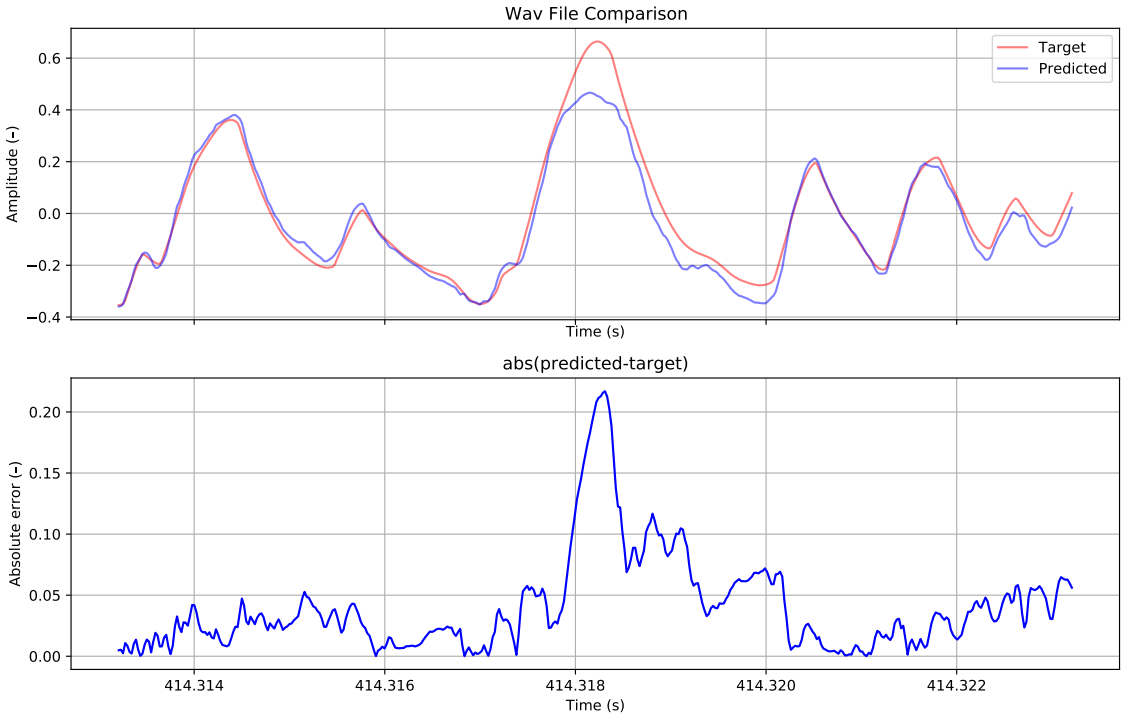
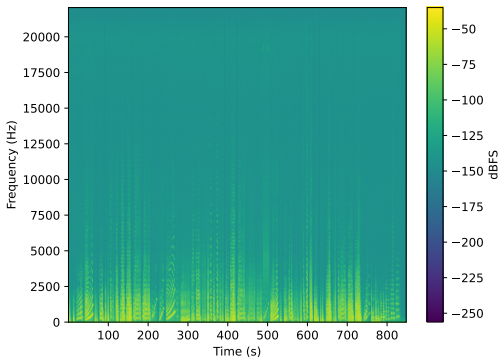


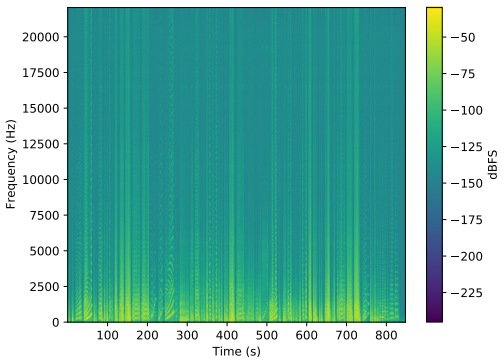
Fig. 5.10: Detailed waveform and absolute error plots of the WaveNet-style *ice* model.

5.1.3 Wem Clubman MK8 5W 1x10

This audio device was expected to be one of the easier ones in terms of modelling since no (intentional) distortion or overdrive was involved. The question in theory was how likely is the neural network to reproduce the *tone* of a tube preamplifier. The models are not only modelling the sound device itself, but also the used microphone and its characteristic and the impulse response of the space where the reamplification has taken place. The predictions of the feedforward model (Fig. 5.12, 5.14) were almost spot on, especially when compared to the WaveNet-style network model (Fig. 5.13, 5.15) that was surprisingly struggling with this type of an audio effect and having trouble modelling faithfully this relatively common waveform. Difference spectrograms of the both models are provided in Fig. 5.11.



(a) Difference spectrogram of the *predicted* signal made by the feedforward *wem* model and the corresponding *target* signal.



(b) Difference spectrogram of the *predicted* signal made by the WaveNet-style *wem* model and the corresponding *target* signal.

Fig. 5.11: Difference spectrograms of the *wem* models provided by the feedforward network (a) and the WaveNet-style network (b).

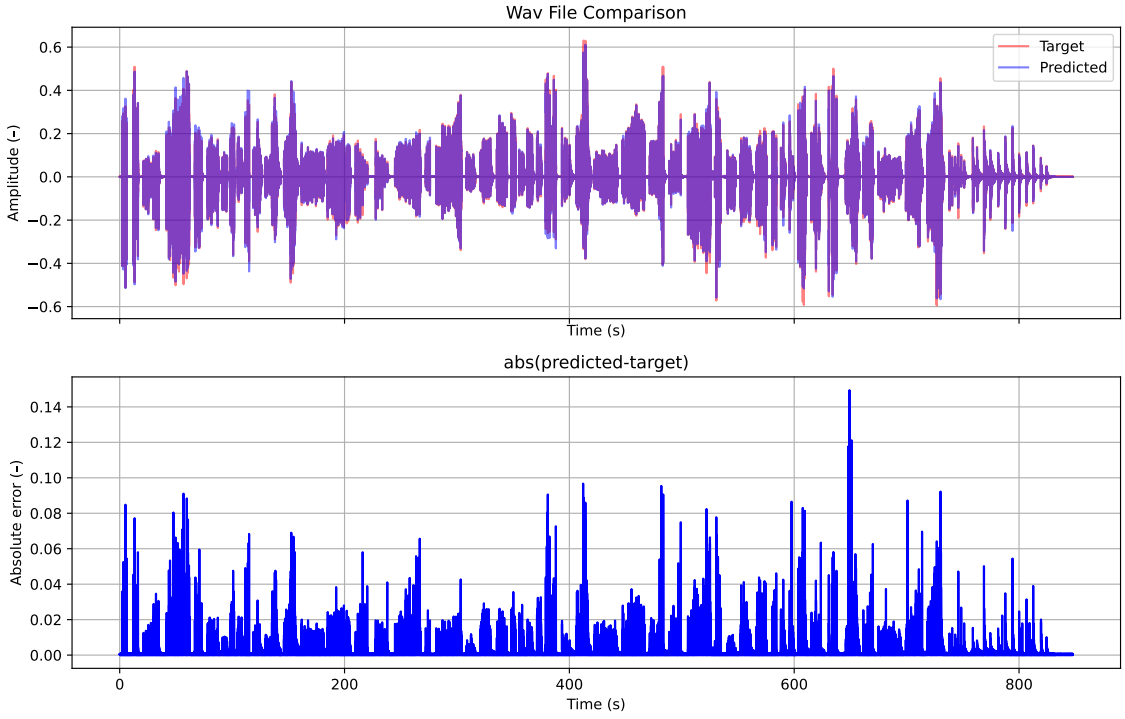


Fig. 5.12: Waveform and absolute error plots of the feedforward *wem* model.

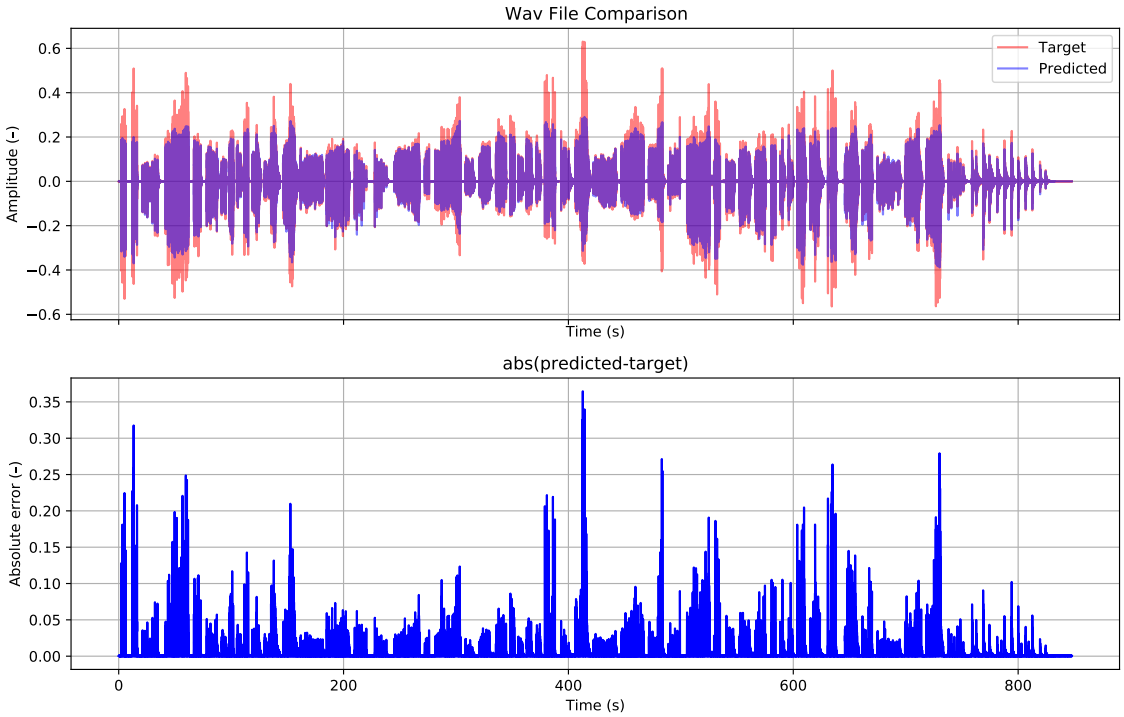


Fig. 5.13: Waveform and absolute error plots of the WaveNet-style *wem* model.

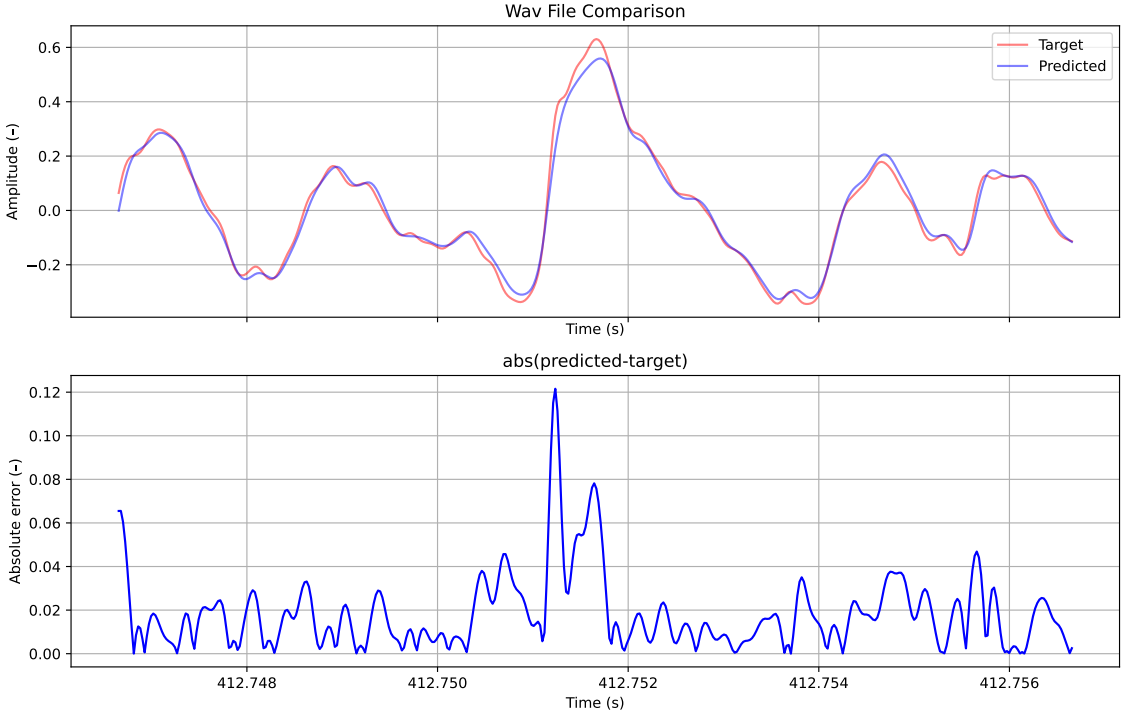


Fig. 5.14: Detailed waveform and absolute error plots of the feedforward *wem* model.

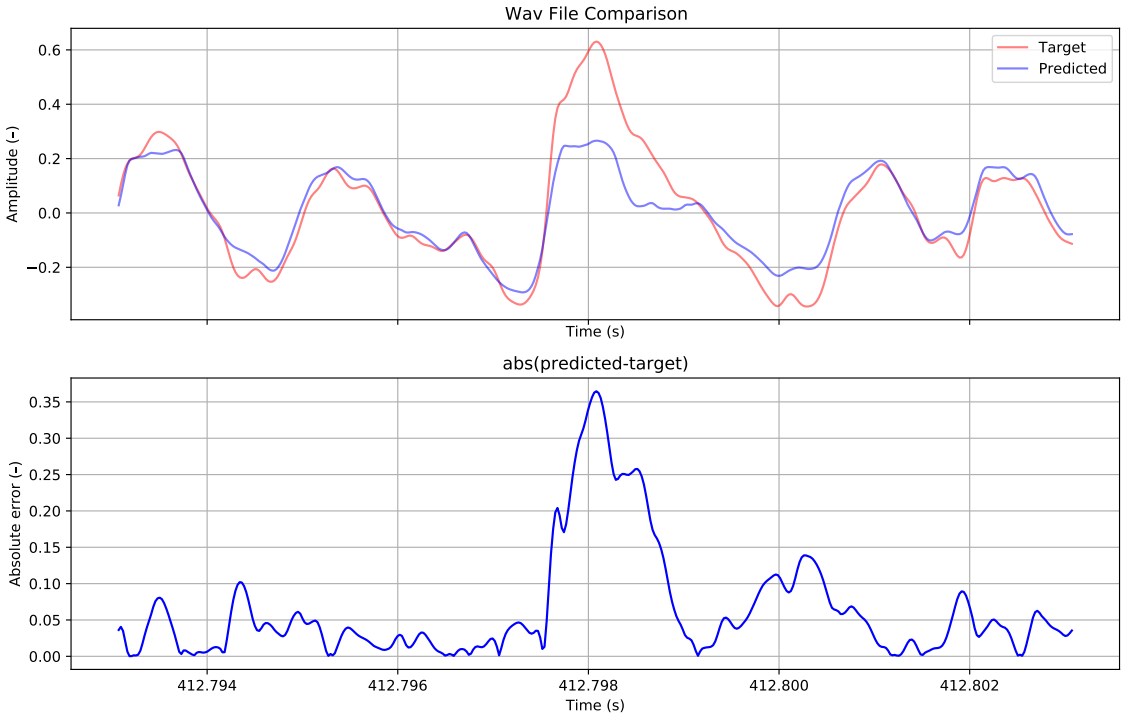
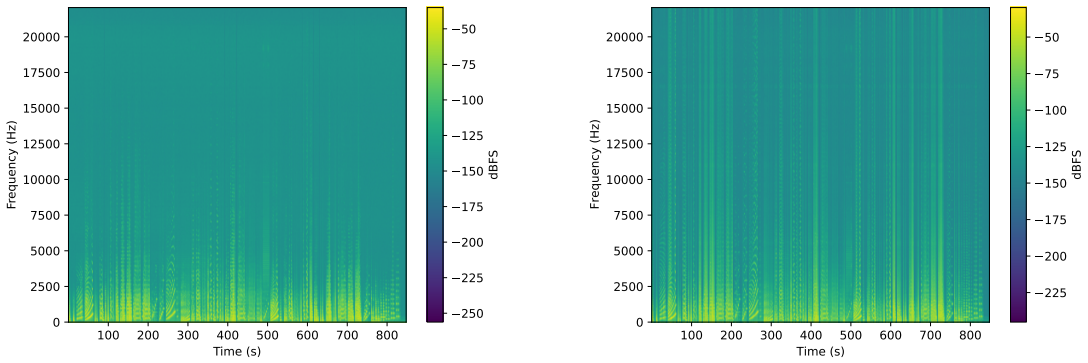


Fig. 5.15: Detailed waveform and absolute error plots of the WaveNet-style *wem* model.

5.1.4 TC Electronic Flashback Triple Delay

The TC Electronic Flashback Triple Delay was the only delay sound effect in the effect pool. The results have proved that the models are capable of recreation of the audio device fairly successfully (see Fig. 5.17 for the waveforms comparison, and 5.19 for the detailed waveforms comparison), including a faithful delay features of the simulated effect. The performance of the feedforward network surpassed the WaveNet-style network, since there are quite a major imprecisions in the WaveNet-style model predictions, as visible in Fig. 5.18, 5.20 respectively. Difference spectrograms available in the Fig. 5.16.



(a) Difference spectrogram of the *predicted* signal made by the feedforward *x4* model and the corresponding *target* signal.

(b) Difference spectrogram of the *predicted* signal made by the WaveNet-style *x4* model and the corresponding *target* signal.

Fig. 5.16: Difference spectrograms of the *x4* models provided by the feedforward network (a) and the WaveNet-style network (b).

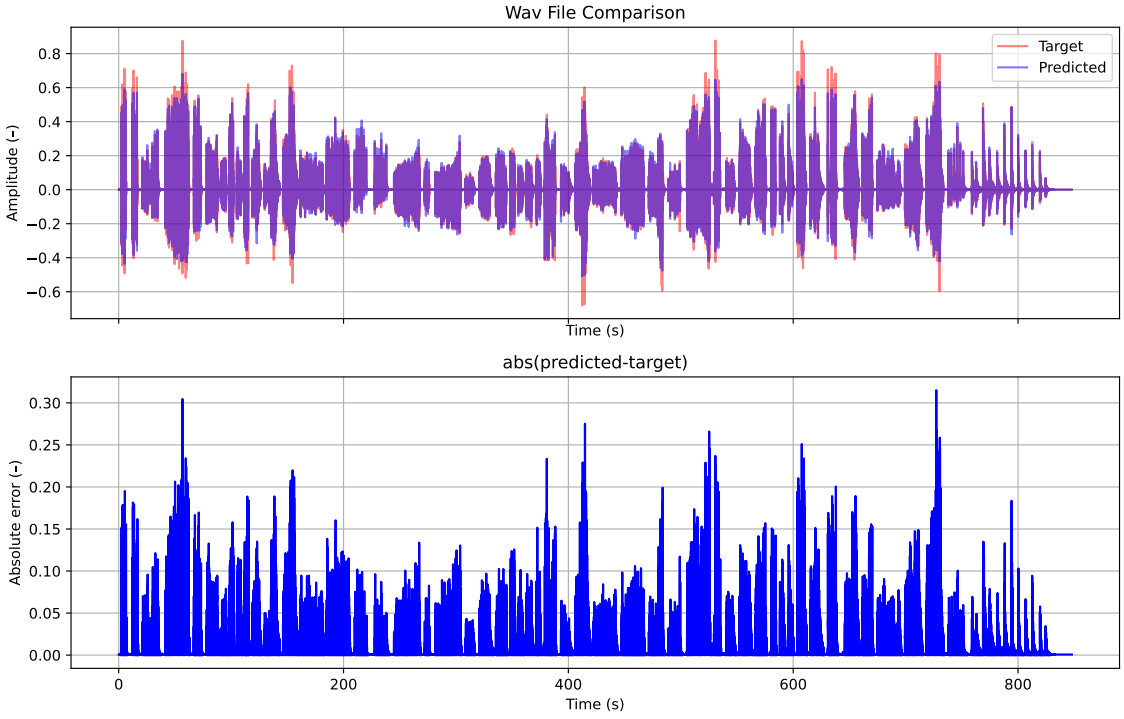


Fig. 5.17: Waveform and absolute error plots of the feedforward *x4* model.

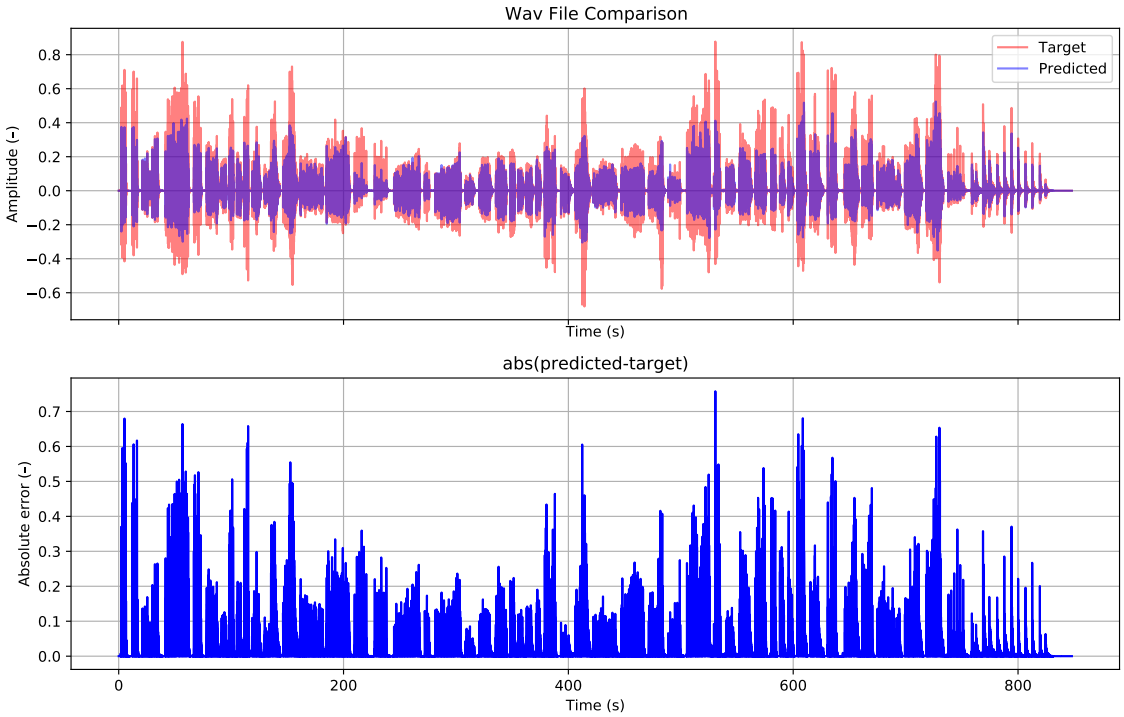


Fig. 5.18: Waveform and absolute error plots of the WaveNet-style *x4* model.

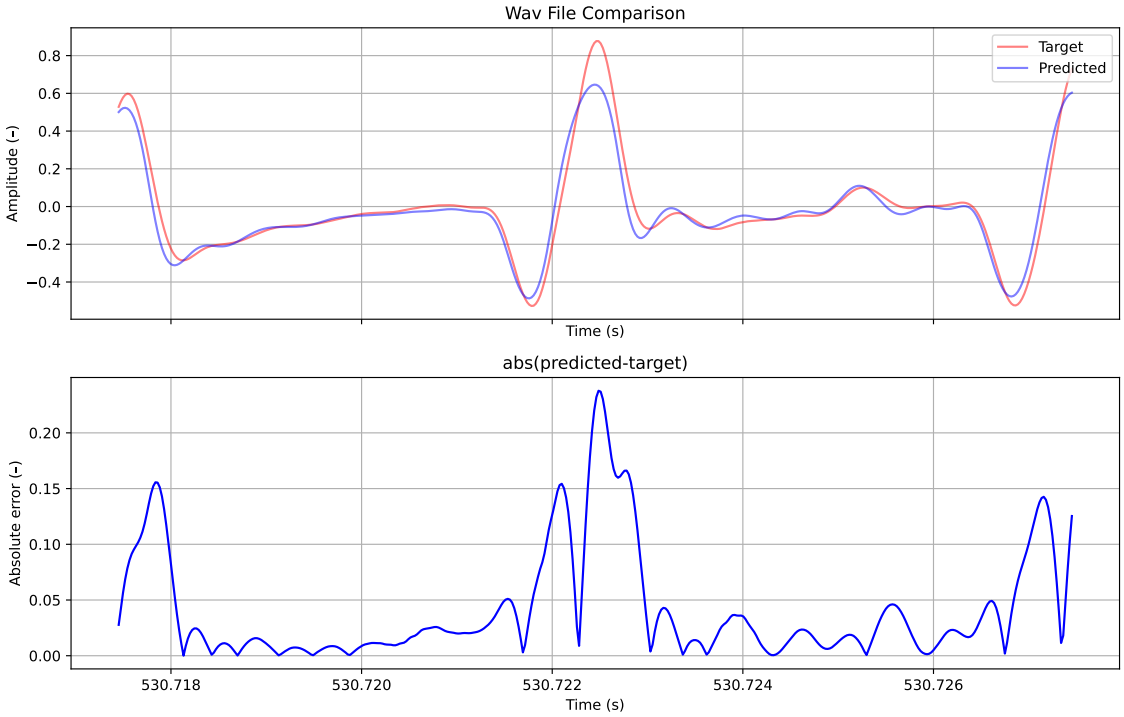


Fig. 5.19: Detailed waveform and absolute error plots of the feedforward *x4* model.

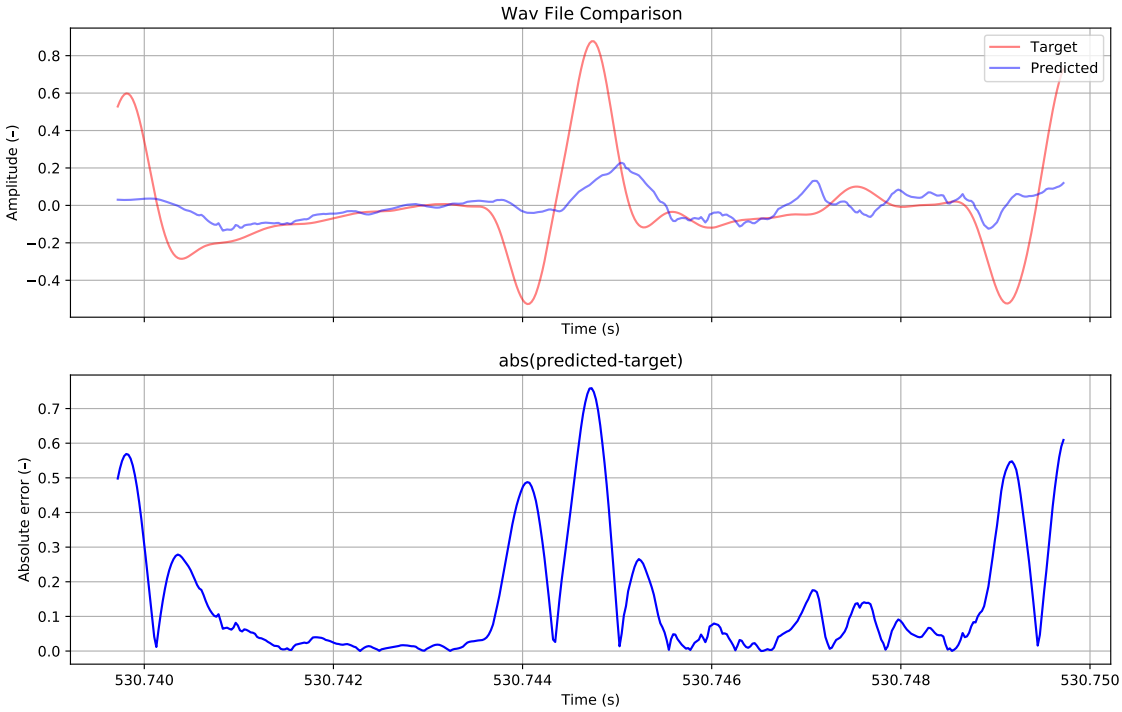


Fig. 5.20: Detailed waveform and absolute error plots of the WaveNet-style *x4* model.

5.2 Qualitative Results

This chapter contains the qualitative results obtained from the MUSHRA listening test (Fig. 5.22), which is done as an extension of the mandatory master thesis assignment.

5.2.1 Listening Test

A listening test for the subjective quality evaluation was created. The test is inspired by the MUSHRA[1] methodology. Network predictions are presented to the test participants as conditions (stimuli). As a reference an audio file excerpt from the original target validation set per effect is used. Same applies to hidden reference. As an anchor, a reference sample that is intentionally degraded in quality is used. This method should ensure that the evaluation scale is calibrated.
Participants mark the conditions on a 1–100 scale. The test interface along with the mark scale can be seen in the Fig. 5.21.

[1]MUltiple Stimuli with Hidden Reference and Anchor

Fig. 5.21: Interface of the webMUSHRA.

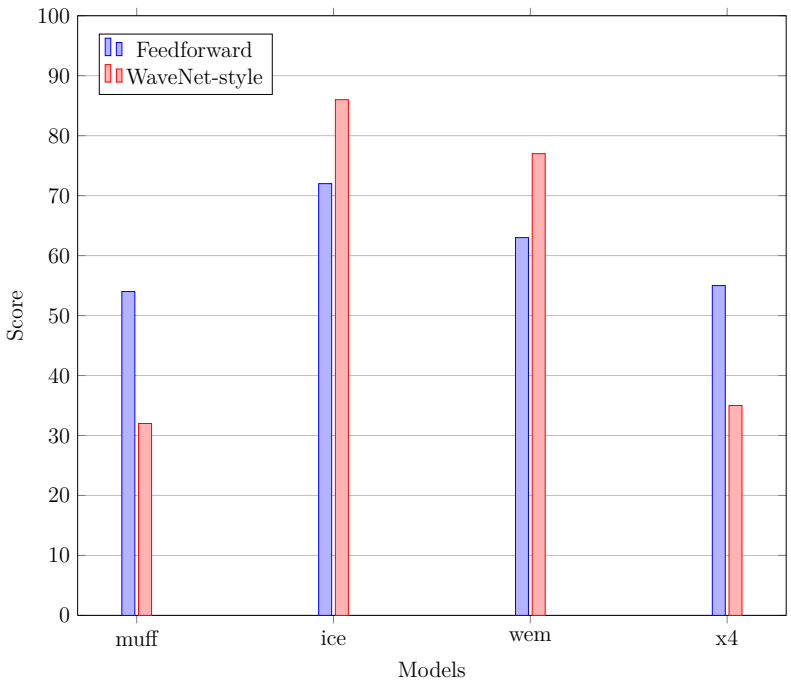


Fig. 5.22: Results of the listening test. Total number of participants: 8.

When the condition is marked 100 (*Excellent*) it means that it is virtually indistinguishable from the provided reference and it sounds identically to the participant. Alternatively conditions marked 0 (*Bad*) doesn't sound like the reference at all.

Testing was carried out via internet questionnaire that uses the web audio API based experiment software webMUSHRA[2] as introduced in [41].

Limitations

When deploying a listening test, one strives for as constant conditions as possible. Normally, this is achieved by e.g. using one type of earphones, DAC's for all participants. Because of the Covid-19 pandemic situation in the Czech Republic as of May, 2021, this requirement could not be satisfied. Such limitations should be taken into account as a another factor influencing the test participants and possibly degrading the results of the listening test in a way.

[2] `https://github.com/audiolabs/webMUSHRA`

Conclusion

The goal of this work was to sum up and organize knowledge and information concerning the topic of audio signal modelling using neural networks. In Chapter 1, types of audio effects were introduced along with different audio modelling techniques, including audio modelling using neural networks.

Chapter 2 mapped historical roots of artificial neural networks, explained basics of the neural network and machine learning field. Several neural network architecture types were described, considering the frequency of use of such architectures in the audio modelling field. Concluding this chapter, strategies of neural network training were discussed.

Main focus of Chapter 3 was the original WaveNet algorithm, which is a relatively recent attempt in the audio modelling field, capable of modelling audio signals directly at the waveform level. Main mechanics of this architecture were described.

In Chapter 4 the used audio effects were introduced, data and its preprocessing were described. Model structures in question were introduced and described as well, same as the training process of the neural network models.

In Chapter 5 results of the experimentation with the best implemented models, that were cautiously picked from over 50 experimental network settings, were displayed and discussed. The chapter is divided into two blocks, evaluating the results of each modelled effect from the both quantitative and qualitative standpoint.

Both used neural networks are far from providing the perfectly performing models, let aside the state-of-the-art black box simulations. Still, in the scope of this thesis the results are feasible, especially when taken in account the limiting factors of time and computational resources. Some models provide a faithful simulation with a strong resemblance of the original audio device. This notion was backed up by the results of the online listening test. Even though some models performed better from the quantitative point of view they were not picked as the better sounding ones by the participants of the listening test as it was the case of *ice* or *wem* models. Previously mentioned limitations of the listening test should also be considered.

Direct comparison of the feedforward neural network and the WaveNet-style neural network cannot be conducted because of the different loss function criterion used per each network. Even though their predictions mostly stand quite closely to each other and the feedforward network maintained overall higher scores in the listening test. The fact that the feedforward network operated with a substantially larger number of trainable parameters (see Appendix A.1) than the WaveNet-style one, might be a possible explanation for its better performance. Real time implementation capability of the feedforward network was examined no further than a rough estimation of the modelling performance, as this was not the scope of this work.

While the original WaveNet algorithm and its variants are considered a powerful model for the human speech synthesis, music synthesis etc., a notion whether it might be both overpowered and yet unnecessarily computationally costly for the task of black box modelling might be raised. All that despite the fact the used PedalNetRT variant of the WaveNet is already considered to be streamlined in terms of computational cost and with black box modelling of audio signals in mind.
As a further extension of this work several improvements might be proposed such as upscaled network structures, more hyperparameters used, usage of the same loss function criterion, more neural network architectures, and effects used, etc.

Bibliography

[1] WRIGHT, Alec, Eero-Pekka DAMSKÄGG, Lauri JUVELA a Vesa VÄLIMÄKI. *Real-Time Guitar Amplifier Emulation with Deep Learning.* Applied Sciences [online]. 2020, 10(3) [cit. 2020-09-14]. DOI: 10.3390/app10030766. ISSN 2076-3417. Dostupné z: <https://www.mdpi.com/2076-3417/10/3/766>

[2] MARTINEZ RAMIREZ, Marco A. a Joshua D. REISS. *Modeling Nonlinear Audio Effects with End-to-end Deep Neural Networks.* In: ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) [online]. IEEE, 2019, 2019, s. 171-175 [cit. 2020-11-14]. ISBN 978-1-4799-8131-1. Dostupné z: <https://doi.org/10.1109/ICASSP.2019.8683529>

[3] MARTÍNEZ RAMÍREZ, Marco A., Emmanouil BENETOS a Joshua D. REISS. *Deep Learning for Black-Box Modeling of Audio Effects.* Applied Sciences [online]. 2020, 10(2) [cit. 2020-09-14]. DOI: 10.3390/app10020638. ISSN 2076-3417. Dostupné z: <https://www.mdpi.com/2076-3417/10/2/638>

[4] SMITH, Julius O. *Physical Audio Signal Processing: for Virtual Musical Instruments and Digital Audio Effects.* Lexington: W3K Publishing, 2010. ISBN 978-0-9745607-2-4.

[5] ZÖLZER, Udo, ed. *DAFX: Digital Audio Effects* [online]. Chichester, UK: John Wiley, 2011 [cit. 2020-11-15]. ISBN 9781119991298. Dostupné z: <https://onlinelibrary.wiley.com/doi/book/10.1002/9781119991298>

[6] PUCKETTE, Miller. *The theory and technique of electronic music.* Hackensack, NJ: World Scientific Publishing, c2007. ISBN 978-981-270-077-3.

[7] REISS, Joshua D. a Andrew MCPHERSON. *Audio Effects: Theory, Implementation and Application* [online]. CRC Press, 2014 [cit. 2020-11-15]. ISBN 9780429097232. Dostupné z: <https://doi.org/10.1201/b17593>

[8] PAKARINEN, Jyri a David T. YEH. *A Review of Digital Techniques for Modeling Vacuum-Tube Guitar Amplifiers.* Computer Music Journal [online]. MIT Press, 2009/05/20, 33(2), 85-100 [cit. 2020-11-15]. ISSN 0148-9267. Dostupné z: <https://doi.org/10.1162/comj.2009.33.2.85>

[9] EICHAS, Felix, Etienne GERAT a Udo ZÖLZER. *Virtual Analog Modeling of Dynamic Range Compression Systems.* Journal of the Audio Engineering Society. 2017, 2017(142), 9752-9752. Dostupné z: <http://www.aes.org/e-lib/browse.cfm?elib=18628>

[10] GERAT, Etienne, Felix EICHAS a Udo ZÖLZER. *Virtual Analog Modeling of a UREI 1176LN Dynamic Range Control System.* Journal of the Audio Engineering Society. 2017, 2017(143), 9852-9852. Dostupné z: <http://www.aes.org/e-lib/browse.cfm?elib=19249>

[11] MÖLLER, Stephan, Martin GROMOWSKI a Udo ZÖLZER. *A Measurement Technique for Highly Nonlinear Transfer Functions.* [online]. 2002/11/15, 2002 [cit. 2020-11-15]. Dostupné z: <https://www.dafx.de/papers/DAFX02_Moeller_Gromowski_Zoelzer_measurement_nonlinear.pdf>

[12] HÉLIE, Thomas. *On the use of Volterra series for real-time simulations of weakly nonlinear analog audio devices: application to the Moog ladder filter.* In Proceedings of DAFx - International Conference On Digital Audio Effects. 2006, 2006, 7-12. Dostupné z: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.422.5228>

[13] EICHAS, Felix a Udo ZÖLZER. *Virtual Analog Modeling of Guitar Amplifiers with Wiener-Hammerstein Models.* In Proceedings of the 44th Annual Convention on Acoustics (DAGA 2018). 2018, 2018. Dostupné z: <https://www.researchgate.net/publication/324720061>

[14] YEH, D. T. *Automated Physical Modeling of Nonlinear Audio Circuits for Real-Time Audio Effects—Part II: BJT and Vacuum Tube Examples.* IEEE Transactions on Audio, Speech, and Language Processing [online]. 2012, 20(4), 1207-1216 [cit. 2020-11-29]. ISSN 1558-7916. Dostupné z: <https://doi.org/10.1109/TASL.2011.2173677>

[15] DE SANCTIS, Giovanni a Augusto SARTI. *Virtual Analog Modeling in the Wave-Digital Domain.* IEEE Transactions on Audio, Speech, and Language Processing [online]. 2010, 18(4), 715-727 [cit. 2020-11-29]. ISSN 1558-7916. Dostupné z: <https://doi.org/10.1109/TASL.2009.2033637>

[16] CHOW, Tommy W S a Siu-Yeung CHO. *A Measurement Neural Networks and Computing.* [online]. IMPERIAL COLLEGE PRESS, 2007 [cit. 2020-11-18]. Series in Electrical and Computer Engineering. ISBN 978-1-86094-758-2. Dostupné z: <https://doi.org/10.1142/p487>

[17] ROSENBLATT, Frank. *The perceptron: A probabilistic model for information storage and organization in the brain.* Psychological Review [online]. 1958, 65(6), 386-408 [cit. 2020-11-29]. ISSN 1939-1471. Dostupné z: <https://doi.org/10.1037/h0042519>

[18] OLGAC, A a Bekir KARLIK. *Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks.* International Journal of Artificial Intelligence And Expert Systems. 2011/02/08, 1, 111-122. Dostupné z: <https://www.researchgate.net/publication/228813985_Performance_Analysis_of_Various_Activation_Functions_in_Generalized_MLP_Architectures_of_Neural_Networks>

[19] *CS231n: Convolutional Neural Networks for Visual Recognition* [online]. Stanford: Stanford University, 2020 [cit. 2020-11-22]. Dostupné z: <https://cs231n.github.io/neural-networks-1/>

[20] KRIZHEVSKY, Alex, Ilya SUTSKEVER a Geoffrey E. HINTON. *ImageNet classification with deep convolutional neural networks.* Communications of the ACM [online]. 2017, 60(6), 84-90 [cit. 2020-11-22]. ISSN 0001-0782. Dostupné z: <https://doi.org/10.1145/3065386>

[21] LECUN, Yann, Yoshua BENGIO a Geoffrey HINTON. *Deep learning.* Nature [online]. 2015, 521(7553), 436-444 [cit. 2020-11-25]. ISSN 0028-0836. Dostupné z: <http://doi.org/10.1038/nature14539>

[22] GOODFELLOW, Ian, Yoshua BENGIO a Aaron COURVILLE. *Deep Learning*[online]. MIT Press, 2016, 1-716 [cit. 2020-11-23]. Dostupné z: <http://www.deeplearningbook.org>

[23] KINGMA, Diederik P. a Jimmy Lei BA. *Adam: A Method for Stochastic Optimization.* Published as a conference paper at ICLR 2015 [online]. 2015, 2015, 1-15 [cit. 2020-11-27]. Dostupné z: <https://arxiv.org/pdf/1412.6980.pdf>

[24] PIOTROWSKI, Adam P. a Jarosław J. NAPIORKOWSKI. *A comparison of methods to avoid overfitting in neural networks training in the case of catchment runoff modelling.* Journal of Hydrology [online]. 2013, 476, 97-111 [cit. 2020-11-28]. ISSN 00221694. Dostupné z: <https://doi.org/10.1016/j.jhydrol.2012.10.019>

[25] SRIVASTAVA, Nitish, Geoffrey HINTON, Alex KRIZHEVSKY, Ilya SUTSKEVER a Ruslan SALAKHUTDINOV. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting.* Journal of Machine Learning Research [online]. 2014/06/01, 2014(15), 1929-1958 [cit. 2020-11-27]. Dostupné

z: <https://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf>

[26] NIELSEN, Michael A. *Neural Networks and Deep Learning* [online]. Determination Press, 2015 [cit. 2020-11-21]. Dostupné z: <https://static.latexstudio.net/article/2018/0912/neuralnetworksanddeeplearning.pdf>

[27] HOCHREITER, Sepp a Jürgen SCHMIDHUBER. *Long Short-Term Memory.* Neural Computation [online]. 1997, 9(8), 1735-1780 [cit. 2020-11-28]. ISSN 0899-7667. Dostupné z: <https://doi.org/10.1162/neco.1997.9.8.1735>

[28] CHO, Kyunghyun, Bart VAN MERRIENBOER, Caglar GULCEHRE, Dzmitry BAHDANAU, Fethi BOUGARES, Holger SCHWENK a Yoshua BENGIO. *Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation.* In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) [online]. Stroudsburg, PA, USA: Association for Computational Linguistics, 2014, 2014, s. 1724-1734 [cit. 2020-11-29]. Dostupné z: <https://doi.org/10.3115/v1/D14-1179>

[29] FUJINO, Akinori, Naonori UEDA a Kazumi SAITO. *A Hybrid Generative/Discriminative Approach to Semi-supervised Classifier Design.* NTT Communication Science Laboratories [online]. Kyoto, Japan, 2005, 764-769 [cit. 2020-11-30]. Dostupné z: <https://www.aaai.org/Papers/AAAI/2005/AAAI05-120.pdf>

[30] ZHU, Xiaojin. *Semi-Supervised Learning Literature Survey.* Computer Sciences TR 1530 [online]. University of Wisconsin – Madison, 2008, 2008, 3-44 [cit. 2020-11-30]. Dostupné z: <http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf>

[31] VAN DEN OORD, Aäron, Sander DIELEMAN, Heiga ZEN, et al. *WAVENET: A GENERATIVE MODEL FOR RAW AUDIO.* ArXiv preprint [online]. 2016, 2016(1609.03499), 1-15 [cit. 2020-11-30]. Dostupné z: <https://arxiv.org/pdf/1609.03499.pdf>

[32] OORD, Aaron van den, Nal KALCHBRENNER, Oriol VINYALS, Lasse ESPEHOLT, Alex GRAVES a Koray KAVUKCUOGLU. *Conditional Image Generation with PixelCNN Decoders.* ArXiv:1606.05328 [cs] [online]. 2016, 2016, 1-13 [cit. 2020-11-30]. Dostupné z: <https://arxiv.org/abs/1606.05328v2>

[33] HE, Kaiming, Xiangyu ZHANG, Shaoqing REN a Jian SUN. *Deep Residual Learning for Image Recognition.* In: 2016 IEEE Conference on Computer Vision

and Pattern Recognition (CVPR) [online]. IEEE, 2016, 2016, s. 770-778 [cit. 2020-12-03]. ISBN 978-1-4673-8851-1. Dostupné z: <https:www.doi.org/10.1109/CVPR.2016.90>

[34] RETHAGE, Dario, Jordi PONS a Xavier SERRA. *A Wavenet for Speech Denoising.* In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) [online]. IEEE, 2018, 2018, s. 5069-5073 [cit. 2020-12-03]. ISBN 978-1-5386-4658-8. Dostupné z: <https:www.doi.org/10.1109/ICASSP.2018.8462417>

[35] DAMSKAGG, Eero-Pekka, Lauri JUVELA, Etienne THUILLIER a Vesa VALIMAKI. *Deep Learning for Tube Amplifier Emulation.* In: ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) [online]. IEEE, 2019, 2019, s. 471-475 [cit. 2020-12-03]. ISBN 978-1-4799-8131-1. Dostupné z: <https:www.doi.org/10.1109/ICASSP.2019.8682805>

[36] KEHLING, Christian, Jakob ABEßER, Christian DITTMAR; Gerald SCHULLER. *Automatic Tablature Transcription of Electric Guitar Recordings by Estimation of Score- and Instrument-related Parameters.* In: Proceedings of the 17th International Conference on Digital Audio Effects (DAFx, 2014), Erlangen, Germany. Dostupné z: <https://www.idmt.fraunhofer.de/en/business_units/m2d/smt/guitar.html>

[37] PROAKIS, John G. a Dimitris G. MANOLAKIS. *Digital signal processing: principles, algorithms, and applications.* 3rd ed. Upper Saddle River, N.J.: Prentice Hall, 1996. ISBN 978-0-13-373762-2.

[38] *ECC83-TK Tube Data Sheet* [online]. 2017. arXiv:1611.09482. Dostupné z: <https://www.telefunken-elektroakustik.com/wp-content/uploads/2017/06/ECC83-TK-Tube-Data-Sheet.pdf>

[39] LE PAINE, Tom, Pooya KHORRAMI, Shiyu CHANG, Yang ZHANG, Prajit RAMACHANDRAN, Mark A. HASEGAWA-JOHNSON a Thomas S. HUANG. *Fast Wavenet Generation Algorithm.* [online]. 2016, , 1-6. arXiv:1611.09482. Dostupné z: <https://arxiv.org/abs/1611.09482>

[40] MURPHY, Kevin P. *Machine learning: a probabilistic perspective.* Cambridge: MIT Press, c2012. Adaptive computation and machine learning series. ISBN 978-0262018029.

[41] SCHOEFFLER, M et al. *webMUSHRA — A Comprehensive Framework for Web-based Listening Tests.* Journal of Open Research Software. 2018 6(1), p.8.

[42] KOKER, Teddy. *Deep Learning for Guitar Effect Emulation.* Teddy Koker [online]. 2020, 10 May 2020. Dostupné z: <https://teddykoker.com/2020/05/deep-learning-for-guitar-effect-emulation/>

Symbols and abbreviations

NN Neural Network

ANN Artificial Neural Network

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

LSTM Long Short-Term Memory

GRU Gated Recurrent Unit

RBM Restricted Boltzmann Machines

CNN Convolutional Neural Network

Softmax Softmax non-linearity

DAW Digital Audio Workstation

MSE Mean Squared Error

ESR Error to Signal Ratio

TCN Temporal Convolutional Network

MUSHRA MUltiple Stimuli with Hidden Reference and Anchor

A Appendix

Architecture	Effect Code-name	Best Validation Loss (MSE)	Effective Batch Size	Input Context	Network Structure	Total Trained Parameters
FF	muff	0.00996	1500000	1024	1024-128-1	131329
FF	muff	0.00983	1500000	1024	1024-256-1	262657
FF	muff	0.00956	2250000	1024	1024-256-256-1	328449
FF	muff	0.00874	1500000	1024	1024-512-256-1	656385
FF	muff	0.00903	1500000	1024	1024-256-256-1	328449
FF	muff	0.00827	1500000	1024	1024-512-512-256-1	919041
FF	muff	0.00763	750000	1024	1024-512-512-256-1	919041
FF	muff	0.00743	450000	1024	1024-512-512-256-1	919041
FF	muff	0.00741	150000	1024	1024-512-512-256-1	919041
FF	ice	0.00033	1500000	1024	1024-512-512-256-1	919041
FF	ice	0.00014	750000	1024	1024-512-512-256-1	919041
FF	**ice**	**0.00011**	**750000**	**2048**	**2048-1024-512-256-1**	**2754561**
FF	wem2	1.88060e-05	1500000	1024	1024-512-512-256-1	919041
FF	wem2	1.70302e-05	750000	1024	1024-512-512-256-1	919041
FF	**wem2**	**1.40679e-05**	**750000**	**2048**	**2048-1024-512-256-1**	**2754561**
FF	**x4**	**0.00040**	**1500000**	**1024**	**1024-512-512-256-1**	**919041**
FF	x4	0.00042	750000	1024	1024-512-512-256-1	919041
FF	x4	0.00043	750000	2048	2048-1024-512-256-1	2754561
FF	muff	0.00946	3000000	2048	2048-256-256-1	590593
FF	muff	0.00853	1500000	2048	2048-256-256-1	590593
FF	muff	0.00915	3000000	2048	2048-512-256-1	1180673
FF	muff	0.00840	1500000	2048	2048-512-256-1	1180673
FF	muff	0.00793	750000	2048	2048-512-256-1	1180673
FF	muff	0.00766	1500000	2048	2048-1024-512-256-1	2754561
FF	**muff**	**0.00735**	**750000**	**2048**	**2048-1024-512-256-1**	**2754561**
FF	muff	0.00747	150000	2048	2048-1024-512-256-1	2754561

Tab. A.1: Full list of the settings used in the training of the feedforward neural network.

Architecture	Effect Code-name	Best Validation Loss (ESR)	Effective Batch Size	Input Context	Network Structure	Total Trained Parameters
PN	ice	0.17587	64	13230	WaveNet-12-10-1-3	10585
PN	ice	0.17193	64	8820	WaveNet-12-10-1-3	10585
PN	**ice**	**0.16888**	**64**	**4410**	**WaveNet-12-10-1-3**	**10585**
PN	ice	0.33128	256	4410	WaveNet-12-10-1-3	10585
PN	ice	0.38486	128	13230	WaveNet-12-10-1-3	10585
PN	ice	0.28278	128	8820	WaveNet-12-10-1-3	10585
PN	ice	0.19245	128	4410	WaveNet-12-10-1-3	10585
PN	**wem**	**0.30842**	**64**	**13230**	**WaveNet-12-10-1-3**	**10585**
PN	wem	0.31608	64	8820	WaveNet-12-10-1-3	10585
PN	wem	0.32677	64	4410	WaveNet-12-10-1-3	10585
PN	wem	0.39240	256	4410	WaveNet-12-10-1-3	10585
PN	wem	0.32899	128	13230	WaveNet-12-10-1-3	10585
PN	wem	0.34606	128	8820	WaveNet-12-10-1-3	10585
PN	wem	0.36496	128	4410	WaveNet-12-10-1-3	10585
PN	x4	0.83214	64	13230	WaveNet-12-10-1-3	10585
PN	x4	0.75176	64	8820	WaveNet-12-10-1-3	10585
PN	**x4**	**0.72024**	**64**	**4410**	**WaveNet-12-10-1-3**	**10585**
PN	x4	0.91977	256	4410	WaveNet-12-10-1-3	10585
PN	x4	0.91161	128	13230	WaveNet-12-10-1-3	10585
PN	x4	0.88955	128	8820	WaveNet-12-10-1-3	10585
PN	x4	0.79665	128	4410	WaveNet-12-10-1-3	10585
PN	muff	0.85871	64	13230	WaveNet-12-10-1-3	10585
PN	muff	0.88910	64	8820	WaveNet-12-10-1-3	10585
PN	**muff**	**0.81137**	**64**	**4410**	**WaveNet-12-10-1-3**	**10585**
PN	muff	0.86202	256	4410	WaveNet-12-10-1-3	10585
PN	muff	0.86890	128	13230	WaveNet-12-10-1-3	10585
PN	muff	0.88865	128	8820	WaveNet-12-10-1-3	10585
PN	muff	0.84217	128	4410	WaveNet-12-10-1-3	10585

Tab. A.2: Full list of the settings used in the training of the WaveNet-style neural network.