BALANCING ERROR AND LATENCY OF BLACK-BOX MODELS FOR AUDIO EFFECTS USING HARDWARE-AWARE NEURAL ARCHITECTURE SEARCH

Christopher Ringhofer, Alexa Gnoss and Gregor Schiele

Intelligent Embedded Systems Lab
University of Duisburg-Essen
Duisburg, Germany
christopher.ringhofer@uni-due.de | alexa.gnoss@stud.uni-due.de | gregor.schiele@uni-due.de

ABSTRACT

In this paper, we address automating and systematizing the process of finding black-box models for virtual analogue audio effects with an optimal balance between error and latency. We introduce a multi-objective optimization approach based on hardware-aware neural architecture search which allows specifying the optimization balance of model error and latency according to the requirements of the application. By using a regularized evolutionary algorithm, it is able to navigate through a huge search space systematically. Additionally, we propose a search space for modelling non-linear dynamic audio effects consisting of over 41 trillion different WaveNet-style architectures. We evaluate its performance and usefulness by yielding highly effective architectures, either up to 18× faster or with a test loss of up to 56% less than the best performing models of the related work, while still showing a favourable trade-off. We can conclude that hardware-aware neural architecture search is a valuable tool that can help researchers and engineers developing virtual analogue models by automating the architecture design and saving time by avoiding manual search and evaluation through trial-and-error.

1. INTRODUCTION

Black-box modelling of virtual analogue audio effects has become an important topic of interest in the digital audio effects community [1, 2, 3, 4, 5, 6, 7, 8] over the last few years. Machine learning methods such as deep neural networks (DNNs) in combination with modern hardware for accelerated training have made it possible to implement data-driven black-box approaches with high quality. A popular approach to implement black-box models is using feed-forward temporal convolutional neural networks (TCNs) [6] or variants like Google DeepMind's WaveNet [9].

However, choosing or designing an appropriate network architecture for a specific audio effect model while considering model quality and latency has shown to be a very tedious and manual process [5]. The architecture of those networks is highly variable in several hyper-parameters such as number of layers, length of convolutional filters per layer, number of channels, activation functions and so on. Hybrid forms of TCNs and WaveNets are also possible. These design choices can have a significant influence on the quality of a black-box model's output but will also impact its latency. In order to find an acceptable balance between model prediction error and inference latency, many different architectures have to be carefully trained and evaluated. It is a difficult effort to systematize this due to the large number of possible architectures in the search space and the interdependence of the various hyper-parameter choices in their impact on model performance.

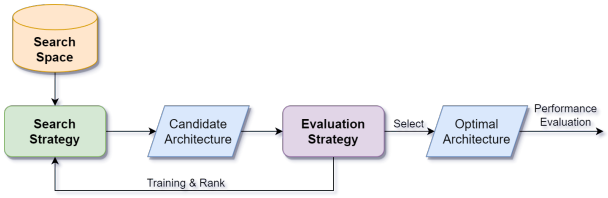

Figure 1: *The general framework of neural architecture search. Adapted from [10]*

With this paper, we propose a solution to this multi-objective optimization problem. We describe our approach based on hardware-aware neural architecture search (HW-NAS) [11] driven by an evolutionary algorithm (EA). Our approach lets users specify the balance of the loss-latency trade-off and relieves them of manually designing and evaluating candidate network architectures according to their quality and latency requirements. We also propose a search space that is suitable for modelling non-linear dynamic audio effects. It offers a wide range of options to explore the loss-latency landscape and arrive at an optimum in reasonable time. Our approach and the search space are evaluated on distortion effects from the IDMT-SMT-Audio-Effects data set [12]. In the evaluation we test the following hypotheses:

1. HW-NAS can automate the design of black-box models for non-linear dynamic audio effects that outperform models from the related work which are designed with other methods.

2. Specifying a balance parameter β in the fitness function leads the evolutionary search to yield either faster or more accurate models.

The remainder of this paper is structured as follows. In Chapter 2 we analyse the related work with respect to systematic design for model accuracy and latency. In Chapter 3 we present our approach and discuss our design decisions for the HW-NAS search space, search strategy and evaluation strategy (cf. Figure 1). We evaluate our system in Chapter 4 and discuss our findings. Chapter 5 concludes our paper. In the end, we discuss meaningful extensions to our approach that will facilitate optimized architecture search for deployment on embedded audio hardware.

2. DESIGN EFFICIENCY AND COMPLEXITY OF VIRTUAL ANALOGUE MODELS

The following chapter discusses different approaches to audio effects modelling and highlights their challenges with respect to design, implementation and achieving a favourable trade-off between model accuracy and computational complexity. Its aim is to understand the shortcomings of existing approaches to facilitate the development of a solution that can address these issues by automating the design of accurate and latency-efficient black-box models.

2.1. White-box modelling

Traditionally, virtual analogue modelling of non-linear audio effect circuits has been performed with white-box models such as wave digital filters, modified nodal analysis or state-space approaches [13]. This requires complete knowledge of the circuit under analysis, meaning the accurate voltage/current relationships at each port need to be known. Acquiring this knowledge may involve tracing circuits, analysing mechanics and physics and performing numerous measurements. The complexity of these approaches increases the more non-linearities are taken into account. In theory, if the circuit is precisely modelled and implemented in the digital domain, the results can sound close to perfect. However, the trade-off between model quality and complexity respective latency needs to be considered and carefully tuned. Thus, in practise, an optimal trade-off is hard to achieve and takes a lot of time and expertise [14]. Due to the complex implications that optimizing for one objective has on the other, this tuning process is difficult to automate [15].

2.2. Grey-box modelling

Grey-box models try to reduce the knowledge required to reproduce the behaviour of an audio effect by employing a partially data-driven approach [16, 17]. This enables such models to integrate information gathered from data with prior knowledge. Usually, only knowledge about the basic circuit topology of an audio effect device is required or may be assumed. Some parameters of those grey-box models are then fitted to input/output data produced with the device using trial and error, grid search or machine learning techniques. For example, simple neural networks may be used to learn a parameter distribution that simulates the behavior of device controls, ultimately integrating black-box sub-models into the overall system model [17]. Another grey-box approach uses differentiable digital signal processing blocks which can be jointly trained end-to-end as layers within a DNN model [18].

The quality and reliability of grey-box models often depends on the extent to which verifiable knowledge of the circuit under analysis is available. If those parts of the model derived from data are not too complex, a grey-box model often remains interpretable. The structure of a grey-box model is typically very specific and only applicable for one class of audio effects, for example, dynamic range compression [17], due to the knowledge and assumptions about the circuit topology included into the design. The complexity of such a model may be adjustable within certain bounds, so that a compromise between model accuracy and computational costs is possible [17]. Due to the nature of grey-box models integrating both white-box and black-box concepts, the task of automating the model design in terms of a trade-off between error and latency consequently involves the challenges of both design approaches.

2.3. Black-box modelling

Black-box modelling sits on the other extreme end of the spectrum and (theoretically) requires no knowledge of the inner workings of the circuit under analysis. The behaviour is modelled solely based on observations of output signals generated when fed with input signals. Usually, black-box models are implemented by deep learning. Mainly, two different classes of DNN architectures have been employed for this task: feed-forward and recurrent neural networks. WaveNets or, more generally, feed-forward TCNs have been used by numerous works in the last few years, for example, to model the non-linear dynamic behaviour of tube amplifiers and distortion circuits [5] or of optical dynamic range compression [6]. Alternatively, recurrent neural networks (RNNs) like long short-term memory (LSTM) networks have been employed for similar applications [3, 4, 8].

Their advantage lies in the potential to find accurate mathematical models which approximate the behaviour of a circuit under analysis solely based on input/output pairs without having to explicitly specify the parameters of a model. Those parameters which yield the most accurate model are automatically determined by training the model with error back-propagation and gradient descent. Moreover, they are conceptually simple, easy to train and to use and can often be processed in parallel [19]. However, the general structure of the mathematical model, that is, the neural network architecture, must be determined a priori. Manual design and fine-tuning of neural network architectures demands expertise and experimentation which consumes time and can be an error-prone process as well, not least due to confirmation biases. Due to the opaque nature of DNN-based black-box models, it is difficult to analyse them or to compute dynamic characteristics from them [20]. Lastly, DNNs require diverse data in large quantities to avoid overfitting which can pose another challenge [19].

2.3.1. Model quality and latency of TCNs and RNNs

In several works both or a combination of WaveNet-style neural networks and recurrent neural networks have been used and compared [1, 4, 5, 6, 7]. When discussing model quality and processing latency, the related work is divided to some extent. Wright et al. [4] evaluated their own C++ implementations of LSTM and WaveNet architectures on a Intel Core i5 2.8 GHz CPU. They measured an 1.5× up to 4.5× faster inference speed for LSTMs compared to the WaveNet architectures while achieving a similar error-to-signal ratio (ESR) when modelling the Blackstar HT-1 tube amplifier and the Electro-Harmonix Big Muff Pi distortion/fuzz pedal, respectively. Steinmetz and Reiss [6], on the other hand, evaluated Python implementations of both TCN and LSTM architectures on a CPU and a GPU, respectively, using the PyTorch deep learning framework. They find their causal TCNs to run up to 37.3× faster than real-time on an Nvidia RTX 3090 GPU and up to 5.0× on a Intel Core i7-8850H 2.6 GHz CPU. On the other hand, although using 32× less parameters than their TCN-300-C architecture, their LSTM network with 32 hidden units achieved 2.8× real-time on a GPU and only 0.9× on a CPU, even when compiled via TorchScript. Moreover, the LSTM required over 8× longer to train until convergence (108 hours) compared to the TCN-300-C model (13 hours).

In these works, a range of architecture configurations is explored and it becomes apparent that their performance is dependent on many factors: the audio effect to model, the efficiency of the implementation of the neural network and its inference run-

time as well as the hardware platform the model is deployed on. To the best of the authors' knowledge, there has not been a systematic study that analyses these factors for TCNs and RNNs in the context of real-time virtual analogue modelling, yet. The most extensive study so far has been done by Wright et al. [5]. They analysed the impact certain hyper-parameter configurations of their WaveNets and LSTMs have on the processing latency and on the ESR, effectively performing a grid search. They employed a fixed WaveNet-style DNN structure and varied the number of layers, the number of convolution channels for the entire network and the activation functions for the entire network (using tanh, ReLU, Gated and SoftGated). They plotted the processing speed as a function of the number of convolution channels with different choices of activation functions or number of layers. For each of the three distortion effects they modelled, they determined an architecture which showed the best trade-off between error and processing speed. The LSTM was varied only in the number of hidden units. Even though various combinations are examined in this paper, a thorough analysis that considers the trade-off between error and latency for other possible hyper-parameter combinations, as well as an automated approach, is missing.

2.3.2. *Challenges*

As outlined above, there are many design decision which influence the performance of a model with regards to error and inference latency. With increased demand for acoustically accurate modelling of analogue audio effects, the requirements on latency become harder to meet. The computational complexity increases with more details and non-linearities of the original analogue effects unit that the digital model is expected to reproduce [5]. Whereas in music studio environments latency of audio effect plugins may be less of a problem when working with pre-recorded audio, live performance scenarios require very low latencies in order to be as non-disruptive to the performing artist as possible. This results in differing requirements for different scenarios: in offline processing environments with sufficient computing power, model accuracy is often favoured at the expense of latency, whereas in online processing environments like live music performances, low latency is of utmost priority, while some sonic details of more accurate models can be neglected.

Albeit powerful in representation capacity, using large DNNs for black-box models demands plenty of time for inference [7].

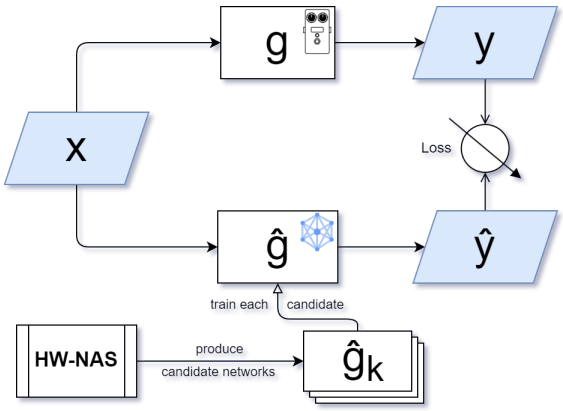

Figure 2: *Overview of proposed black-box modelling system.*

Going for smaller or less complex networks requires less computational effort which has a positive impact on inference latency but potentially affects model accuracy [5, 6]. With the endless options there are to design DNN architectures, the question becomes: *How to choose a design that provides an optimal balance between model accuracy and inference latency?* The main challenge that we want to address with our approach is providing a systematic and automated solution to this question.

3. HW-NAS APPROACH

In this chapter, we want to propose our solution to the aforementioned challenges. We present our approach based on HW-NAS, the search space, the search strategy and the evaluation strategy (cf. Figure 1) and discuss our design decisions.

3.1. System overview

We choose to use HW-NAS to address the challenges mentioned in section 2.3.2 and to enable us to efficiently build dedicated, accurate and time-efficient black-box models for each audio effect.

The relatively new field of neural architecture search (NAS) aims at automating the DNN design process to some extent [10]. It intents to replace the manual effort in determining a suitable model architecture by automated experimentation and manipulation of the architecture, often aided by a separate artificial intelligence like RNNs or EAs. However, NAS solely focuses on maximizing the predictive quality of a DNN without considering the resource consumption of its implementation on a given hardware platform like a GPU or CPU. Furthermore, the optimal DNN architecture for a task and platform can vary significantly depending on what costs must be optimized – be it latency, energy consumption, memory consumption, etc. An extension of NAS called hardware-aware neural architecture search (HW-NAS) attempts to address this problem. It takes estimates (or measurements) of the resource consumption of a particular DNN implementation on a particular hardware platform into account for the automated search [11]. The hardware-aware property allows us to consider hardware-dependent metrics like inference latency in our search.

A NAS system uses a *search strategy* to systematically explore a *search space* describing a very large number of DNN architectures and evaluates them according to an *evaluation strategy*. In our system, part of this evaluation strategy must be to calculate an appropriate error metric for the black-box modelling task at hand and optimize it as a loss function. Figure 2 shows schematically how the HW-NAS system produces candidate DNN architectures $\hat{g}_k$ that are trained by comparing their output $\hat{y}$ to the output y of an audio effect g to model when given the same input x.

In the remainder of this chapter, we will describe how we built our HW-NAS system to accomplish our set goals. Our system has been built in Python with the Retiarii framework by Microsoft's Neural Network Intelligence[1].

3.2. Search space

Our search space is intended to serve as a basis for modelling nonlinear dynamic audio effects such as overdrive, distortion, fuzz and dynamic range compression. We therefore based our design on the causal, feed-forward TCN and WaveNet architectures introduced by Steinmetz et al. [6] and Wright et al. [5].

[1] https://nni.readthedocs.io/en/v2.10/

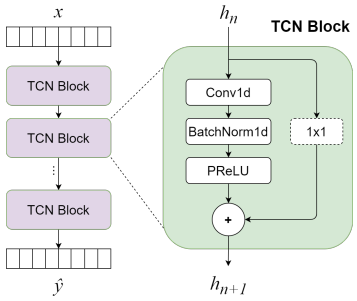

Figure 3: *TCN example structure. Adapted from [6]*

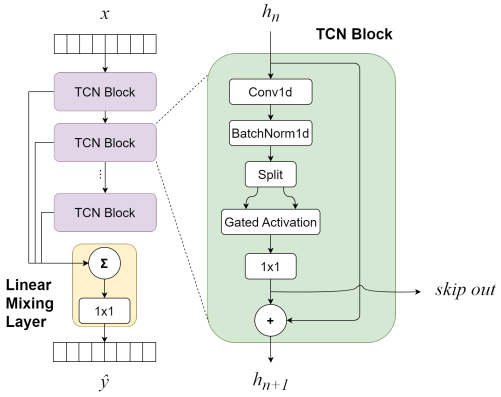

Figure 4: *WaveNet example structure.*

The basic structure of a TCN in the style of Steinmetz et al. is illustrated in Figure 3. It is a sequential structure built from a number of repeated temporal convolutional blocks. Each block consists of a dilated, causal convolution filter (with a fixed channel depth), a batch normalization, a parametrized rectified linear unit (PReLU) and a residual connection. The residual connection is linearly scaled by a 1×1 convolutional kernel with the same channel depth. The dilation factor grows exponentially per layer, depending on its depth and the stack size which resets the exponent. The PReLU activation function is defined in Equation 1, where α is a learnable array with the same shape as x. We removed the conditioning module from the structure introduced by Steinmetz et al., but reintroduced causal padding as implemented in the first version of their paper to ensure the input and output dimensionalities are equal.

$$\mathrm{PReLU}_\alpha(\mathrm{x}) = \begin{cases} \alpha \cdot x & \text{if } x \leq 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{1}$$

$$\mathrm{GatedActivation}(\mathrm{x}, \mathrm{H}) = \tanh\left(H_f * x\right) \cdot \sigma(H_g * x) \tag{2}$$

The basic structure of a WaveNet in the style of Wright et al. is illustrated in Figure 4. It follows a similar sequential structure to the TCN with a few differences. The internal structure of its TCN block replaces the PReLU activation with a gated activation [9] which is defined in Equation 2. H_f and H_g are filter and gate convolution kernels and can each be considered as one half of the convolutional kernel H split along the depth dimension. σ is the logistic sigmoid function generating the gating signal. A final 1×1 convolutional kernel restores the output dimensionality after the gated activation. The output of the WaveNet is not the output of the last TCN block, though. Each block has a skip connection that leads to a linear mixing layer which sums and scales the results of all TCN blocks to produce the final output of the network. The linear mixing layer is taken from Wright et al. [5] and replaces the more complex original non-linear post-processing module [9]. The underlying system model of TCNs and WaveNets can be considered a cascade of Wiener models [5]. Each convolutional filter is a linear time-invariant system with finite impulse response and learnable coefficients, which is followed by a static non-linear activation function to approximate a non-linear stateful filter. Using only causal temporal convolutions limits the model to consider only current and past input values, which conforms to the causal nature of implementable analogue systems. Our layer-wise search space subsumes these basic structures and parameterizes them with the options shown in Table 1 to generate a range of different architectures from TCNs, WaveNets and in between. Each of the seven blocks except the first can be deactivated. In total, this yields 41.231.686.041.600 different hyper-parameter combinations in our search space. Each resulting architecture can be easily built with our generator code. By providing a vast search space with learnable Wiener-style models of different size and complexity, we make very few assumptions about the design of the circuit under analysis, and so should be able to model of a broad range of non-linear dynamic systems.

Table 1: *Hyper-parameter options in our proposed search space.*

Hyper-parameter	Options
Linear mixing layer + skip con.	false, true
Number of blocks	1, 2, 3, 4, 5, 6, 7
Dilation growth	2, 4, 6, 8, 10
Stack size	1, 3, 5, 7
Convolution channels	2, 4, 8, 16, 32
Kernel size *(per block)*	3, 5, 7, 9, 11, 13, 15, 17
Residual convolution *(per block)*	false, true
Activation function *(per block)*	PReLU, Gated

3.3. Search strategy

The search strategy is responsible for exploring and updating the search space to find the "best" architectures. The most commonly used search algorithms in HW-NAS are random search, grid search, evolutionary algorithms (EAs) and reinforcement learning algorithms. In recent years, new upcoming search algorithms are gradient-based methods which require a specially designed search space and relaxations of non-differentiable hardware metrics [11].

With over 41 trillion possible candidates in the search space, more trivial algorithms like random search or grid search are too uniform and provide too little coverage. According to the surveys [10, 11] there is no clear preference between RLs and EAs for HW-NAS. We decided to use an EA for our system with a random initialization phase. After initializing a first population of architectures as a starting point, EAs consist of four steps that are reiterated multiple times:

1. Select parents from the population for reproduction.
2. Apply mutation operations to create new individuals.
3. Evaluate the fitness of the new individuals.
4. Select the survivors of the population.

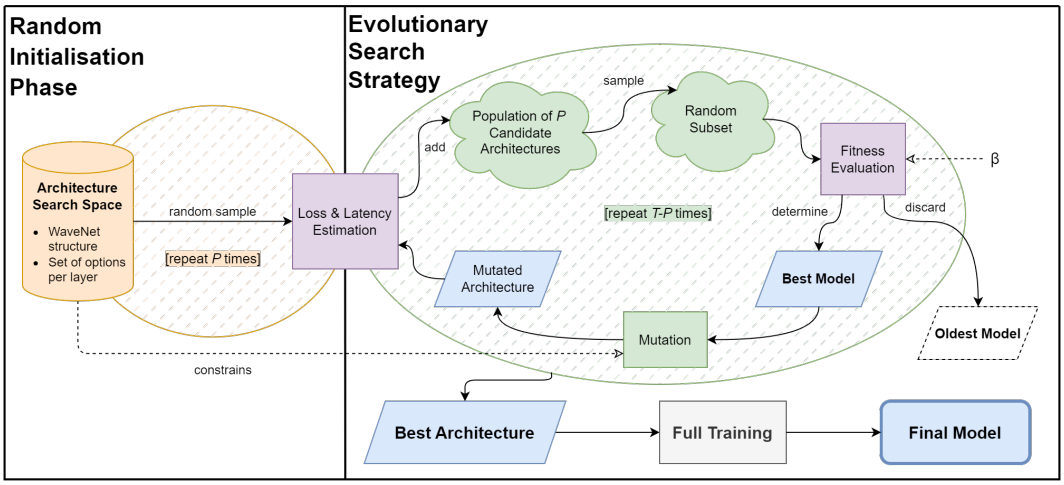

Figure 5: *Overview of the proposed search algorithm. Colour coding analogous to Figure 1.*

An overview of the different phases is illustrated in Figure 5. In the random initialization phase the initial population is created by performing a random search over the entire search space for P trials, where P is the size of the population. We set $P = 150$. Each trial yields one architecture which is then evaluated for its loss and latency which contribute to its overall fitness. This step will be further detailed in the next Section 3.4. For the second phase, we chose to use a slightly modified regularized EA from Real et al. [21] in our search strategy. In the beginning, a random sub-population of 20% is sampled and ranked by their fitness value. Since the regularized EA is an aging EA that introduces an "age property" and keeps track of which architecture is the oldest in the population, the oldest one is discarded. The architecture of the fittest model, the so-called parent architecture, is randomly mutated into a child architecture. The expected value of mutations per iteration is set to $E(p) = 1.5$ with $p = \frac{1.5}{31}$, since 31 adjustable hyper-parameters are available (cf. Table 1). The new architecture is evaluated for loss and latency again and added to the population. The overall procedure lasts for $T = 1000$ trials maximum. This algorithm has the advantage that the random population from the beginning is systematically refined and filtered for its fittest and most promising candidates. Since only 20% of the population are considered in each trial, the algorithm maintains several promising candidates in parallel. After the final trial, the fittest architecture of the entire search history is trained until convergence and yields the final, optimal model.

3.4. Evaluation strategy

The purpose of the evaluation strategy is to rank the candidate architectures according to our specified optimization objectives. Our goal is to find models with an optimal balance between error and latency. In the following, we describe how we acquire these metrics and how we combine them into an optimization objective that is used for the fitness value.

We calculate loss functions based on both time and frequency in accordance with Steinmetz et al. [6]. We use the L_1 loss for the time-domain component $\mathcal{L}_{\text{time}}$ and the multi-resolution Short-Time Fourier Transform error for the frequency-domain component $\mathcal{L}_{\text{freq}}$ [22]. The final loss function can be seen in Equation 3. In contrast to Steinmetz et al. [6], we chose $\alpha = 75$. This value was iteratively determined so that at convergence both components contribute approximately equally to $\mathcal{L}_{\text{overall}}$ and thus neither of the components dominates the term during optimization.

$$\mathcal{L}_{\text{overall}}(y_i, \hat{y}_i) = \mathcal{L}_{\text{time}}(y_i, \hat{y}_i) + \alpha \cdot \mathcal{L}_{\text{freq}}(y_i, \hat{y}_i) \tag{3}$$

In order to optimize the runtime performance of the search, we use two techniques for loss approximation [11]. We employ early stopping on the validation loss with a patience of 10 epochs, $\Delta_{min} = 0.0005$ and a maximum of 50 training epochs. Furthermore, the models are trained only with 16-bit half-precision floating point numbers, whereas they are trained with 32-bit full-precision for the final evaluation. This further speeds up the search while still giving a reasonable approximate for the fitness computation. The latency of the PyTorch implementation is measured directly on the CPU with the help of the torch.profiler package. A randomized input vector resembling 2 seconds of mono audio is prepared and fed into the model once for warming up. Then, the latency is averaged over 25 consecutive inferences to minimize the impact of variance on the measurement.

When optimizing for an optimal trade-off between model error and inference latency, we are looking at a multi-objective optimization problem with model error and inference latency as co-optimization objectives. To simplify the search, we use the scalarization method [11] to combine both objectives into a single fitness function. Specifying a balance parameter β in the fitness function that increases or decreases the impact that either of those objectives have on the fitness function, leads the evolutionary search to yield either faster or more accurate models. The final fitness function f is described by Equation 4, with $\mathcal{L}_{val}$ being the overall loss on the validation data set according to Equation 3 and t_Δ denoting the inference latency in milliseconds. Higher values for β increase the influence that lower losses ($\mathcal{L}_{\text{val}} \ll 0$) have on the fitness.

$$f = \frac{10^\beta}{\mathcal{L}_{\text{val}}^\beta \cdot t_\Delta} \tag{4}$$

Table 2: *Baseline architectures.*

Hyper-parameter	**TCN-100-C** [6]	**TCN-300-C** [6]	**WaveNet1** [5]	**WaveNet2** [5]	**WaveNet3** [5]
Number of blocks	4	4	10	18	18
Linear mixing layer + skip con.	false	false	true	true	true
Dilation growth	10	10	2	2	2
Stack size	4	4	10	9	9
Convolution channels	32	32	16	8	16
Kernel size *(all blocks)*	5	13	3	3	3
Residual convolution *(all blocks)*	true	true	false	false	false
Activation function *(all blocks)*	PReLU	PReLU	Gated	Gated	Gated

4. EXPERIMENTS

In this chapter, we first detail our experiment setup. Then we describe the experiments that we performed, compare our results with the baseline architectures and discuss our findings.

4.1. Experiment setup

4.1.1. Data set

For our evaluation, we used the IDMT-SMT-Audio-Effects data set [12] by the Fraunhofer Institute, which is a popular data set in the DAFx community [2, 4, 5, 7, 16]. It is developed for automatic detection of audio effects in recordings of electric guitar and bass and related signal processing tasks. It provides clean and effected audio data of two different guitars and bass guitars, each with two different pick-up settings and up to three different plucking styles. Considering only the clean, unprocessed audio samples, there are 624 unique bass guitar recordings, 624 unique monophonic guitar recordings and 420 unique polyphonic guitar recordings. The monophonic recordings cover the common pitch range of a 4-string bass guitar respective the common pitch range of a 6-string electric guitar. The polyphonic recordings contain simple triads as well as more complex, dissonant chords. Each recording is a mono WAV file, has a duration of exactly 2 seconds, is sampled with 44.1 kHz and a resolution of 16 bits. The recordings are randomly split into fixed data subsets of 60% for training, 20% for validation and 20% for testing.

Additionally, each of those audio samples is processed with a range of different audio effects. We used the effects coded with "4412" and "4413" for the black-box modelling task. They are digital emulations; "4412" is the "Screaming Distortion" by Cubase and "4413" is the "MDA Overdrive" by MOD Audio. Each effect is applied with fixed control settings. "4412" has its "Drive" control set to 95% and its "Contour" set to 90%, whereas "4413" has its "Drive" set to 40% and its "Muffle" control set to 40%. This results in a harsh, distorted sound for "4412" and a smoother, less distorted sound for "4413". Although digitally emulated, the effects sufficiently serve the purpose of demonstrating the performance of our system and the influence of the balance parameter β.

4.1.2. Hardware

Our training hardware consists of 2 separate servers, hosting 4× and 3× Nvidia Quadro RTX8000 GPUs, respectively. Each GPU has 48 GB of GDDR6 ECC VRAM. We used the GPUs for parallelizing and accelerating the training of the candidate architectures. Each server also hosts 2× AMD EPYC 7252 8C 3.10 GHz CPUs with 256 GB RAM. The CPUs were used to evaluate the models and to measure their inference latency. Both servers run Ubuntu 22.04.4 LTS and the Nvidia CUDA Toolkit 11.7.

4.1.3. Configuration

In order to test our hypotheses, we performed a number of experiments.

For the first hypothesis, we took the TCN-100-C, TCN-300-C, WaveNet1, WaveNet2 and WaveNet3 architectures from the related work as a baseline. TCN-100-C and TCN-300-C have a receptive field of around 100 ms and 300 ms, respectively, and are developed to model the LA-2A optical dynamic range compression unit [6]. TCN-100-C is the smallest and fastest network presented in that work. According to Wright et al. [5], WaveNet1 is the fastest of their models, having the worst ESR on the validation data. WaveNet3 is the slowest model, having best ESR on the validation data. WaveNet2 is an intermediate model. In the original work, all three architectures are used for the simulation of the Ibanez Tube Screamer, the Boss DS-1 and the Electro-Harmonix Big Muff Pi. All architectures are built with our own generator code by instantiating them with the hyper-parameters described in Table 2, in order to eliminate any implementation-dependent effects on their performance and evaluate only the capabilities of the respective architectures. They are trained on the clean input data and the "4412" as well as "4413" target data from the training data set. The models with the lowest loss according to Equation 3 on the validation data set are exported as the best-performing models. The final numbers used for the evaluation are taken from their performance on the test data set.

For the second hypothesis, we performed seven HW-NAS experiments. We performed one set of experiment runs for the modelling of the "Screaming Distortion" and the "MDA Overdrive" each. In each set, we did three experiment runs while choosing the balance parameter of Equation 4 to either one of $\beta \in \{3, 7, 10\}$. With smaller values of β, we expect the HW-NAS to prefer faster models at the expense of accuracy. With higher values of β, we expect the HW-NAS to prefer more accurate models at the expense of inference speed. For the "Screaming Distortion", we performed an additional experiment with $\beta = 2$ to further explore the lower limit regarding latency. We set a budget of 96 GPU hours for training, which translates to a runtime of 24 hours on the server with 4 GPUs and 32 hours on the server with 3 GPUs. All GPUs were used in parallel for training. Those architectures that achieved the highest fitness score of an experiment run were used for the final evaluation. In the final evaluation, each architecture was trained for a maximum of 100 epochs. Early stopping was used on the validation loss with a patience of 10 epochs but with $\Delta_{min} = 0.0001$ which waits until the convergence plateau is reached.

Table 3: *HW-NAS result architectures for different effects and different values of β. "—" denotes a block is not present, i.e., not selected.*

Audio effect		**4412**				**4413**		
Balance		$\beta = 2$	$\beta = 3$	$\beta = 7$	$\beta = 10$	$\beta = 3$	$\beta = 7$	$\beta = 10$
Linear mixing layer + skip con.		false	false	false	true	false	false	false
Dilation growth		4	6	10	6	4	8	8
Stack size		7	7	5	7	7	5	5
Convolution channels		8	8	16	32	2	8	16
TCN block #1	Kernel size	11	11	9	11	13	5	5
	Residual convolution	false	false	true	false	true	true	true
	Activation function	Gated	PReLU	Gated	PReLU	Gated	Gated	Gated
TCN block #2	Kernel size	15	9	15	5	5	13	9
	Residual convolution	false	false	false	false	false	true	false
	Activation function	PReLU	Gated	Gated	Gated	PReLU	PReLU	Gated
TCN block #3	Kernel size	17	11	7	17		9	5
	Residual convolution	true	false	false	true	—	false	true
	Activation function	PReLU	Gated	PReLU	Gated		Gated	Gated
TCN block #4	Kernel size		7	9	9		9	7
	Residual convolution	—	false	true	false	—	false	false
	Activation function		Gated	Gated	Gated		Gated	Gated
TCN block #5	Kernel size		13	9	9		11	13
	Residual convolution	—	false	false	false	—	false	true
	Activation function		Gated	Gated	PReLU		Gated	PReLU
TCN block #6	Kernel size			5	9		11	11
	Residual convolution	—	—	true	false	—	true	true
	Activation function			Gated	PReLU		Gated	PReLU
TCN block #7	Kernel size						5	11
	Residual convolution	—	—	—	—	—	true	false
	Activation function						PReLU	Gated

Table 4: *Test loss and latency comparison. $\mathcal{L}_{\text{test}}$ is computed with Equation 3, t_Δ is measured per inference on 2 seconds of audio.*

Effect	**Metric**	**TCN100C**	**TCN300C**	**WaveNet1**	**WaveNet2**	**WaveNet3**	$\beta = 2$	$\beta = 3$	$\beta = 7$	$\beta = 10$
4412	$\mathcal{L}_{\text{test}}$	0.0092	0.0075	0.0083	0.0087	0.0081	0.0066	0.0050	0.0038	**0.0033**
	$t_\Delta\ [ms]$	276.0	310.4	110.1	117.7	211.9	**26.1**	41.3	100.2	504.1
4413	$\mathcal{L}_{\text{test}}$	0.0073	0.0069	0.0062	0.0060	0.0053	—	0.0129	0.0052	**0.0045**
	$t_\Delta\ [ms]$	261.9	321.4	114.2	108.7	206.8	—	**5.9**	62.6	99.7

4.2. Results

In Table 3 the fittest architectures of each experiment run are shown. It is clearly visible that smaller values for β prefer architectures with less TCN blocks since they introduce additional latency. Similarly, smaller values for β prefer architectures with less convolution channels, whereas higher values lead to a greater number of channels, increasing the size of the filter bank that each causal convolution layer represents. Only for the audio effect "4412", a WaveNet structure with skip connections and a linear mixing layer is chosen. In comparison, residual convolutions are more likely to be chosen for the audio effect "4413".

As can be inferred from Table 4 for the simulation of effect "4412", our fastest model ($\beta = 2$) is 4.2× faster than the fastest model from the related work (WaveNet1) while still achieving a 20% lower loss, being even lower than all selected models from the related work. Our most accurate model ($\beta = 10$) achieves a 56% lower loss than the most accurate model from the related work (TCN-300-C), being 1.6× as slow, while still being around 4× faster than real-time on the AMD EPYC 3.10 GHz CPU. For the simulation of effect "4413", our fastest model ($\beta = 3$) is around 18× faster than the fastest model from the related work (WaveNet2). However, its loss is also about 115% higher. This is by design since we told the evaluation strategy to prefer lower latency and, if needed, sacrifice accuracy. Our most accurate model ($\beta = 10$) achieves a 15% lower loss than the most accurate model of the related work (WaveNet3) while still being about 2× faster, similarly fast to WaveNet2.

In general, all evaluated models can simulate both effects with reasonable accuracy. Listening tests show that all models sound fairly similar. While those models with the lowest loss values produce audio which is virtually indistinguishable from the target, those models with higher loss values tend to over- or undersaturate and produce slight artifacts when presented with hiss or other non-musical noise.

5. CONCLUSIONS

For this paper, we developed an approach to automatically design black-box models for non-linear dynamic audio effects based on hardware-aware neural architecture search. We analysed the related work and demonstrated the need for a systematic approach that addresses the trade-off between model error and latency. We

defined a search space that contains over 41 trillion different TCNs and WaveNet-style architectures and a search strategy that can navigate through this huge search space systematically and goal-oriented. Introducing a balance parameter β in the fitness function allows us to effectively influence said trade-off for the resulting black-box model by putting more weight either on the model error or on its inference latency. The performance of our system was demonstrated in the evaluation. Our automated search found models either up to $18\times$ faster or with a test loss of up to 56% less than the best performing models of the related work, while still showing a favourable trade-off. We hope that our approach can serve as a valuable tool that can help researchers and engineers developing accurate and time-efficient black-box models.

In the future, we want to address embedded target hardware platforms with our approach. By using an automatic deployment pipeline and incorporating a DNN-based inference latency estimator into our evaluation strategy, we can systematically search for architectures that will perform best on other hardware platforms. Currently, we are working on such a system for a Raspberry Pi 4 with Elk Audio OS using WaveNetVA [5] as well as one for embedded field-programmable gate arrays (FPGAs) by AMD/Xilinx. For the latter, we are developing optimized DNN layer templates that can be incorporated into our search space. We also plan to extend our approach to include LSTM-based DNN architectures for the simulation of time-varying audio effects [16].

6. REFERENCES

[1] Will J. Cassidy and Enzo De Sena, "Perceptual Evaluation and Genre-specific Training of Deep Neural Network models of a High-gain Guitar Amplifier," in *Proceedings of the 26th International Conference on Digital Audio Effects*, Copenhagen, Denmark, Sept. 2023.

[2] Riccardo Simionato and Stefano Fasciani, "Deep Learning Conditioned Modeling of Optical Compression," in *Proceedings of the 25th International Conference on Digital Audio Effects*, Vienna, Austria, Sept. 2022.

[3] Aleksi Peussa, Eero-Pekka Damskagg, Thomas Sherson, Stylianos I. Mimilakis, Lauri Juvela, Athanasios Gotsopoulos, and Vesa Valimaki, "Exposure Bias and State Matching in Recurrent Neural Network Virtual Analog Models," in *Proceedings of the 24th International Conference on Digital Audio Effects*, Vienna, Austria, Sept. 2021.

[4] Alec Wright, Eero-Pekka Damskägg, and Vesa Välimäki, "Real-Time Black-Box Modelling With Recurrent Neural Networks," in *Proceedings of the 22nd International Conference on Digital Audio Effects*, University of Birmingham, Sept. 2019.

[5] Alec Wright, Eero-Pekka Damskägg, Lauri Juvela, and Vesa Välimäki, "Real-Time Guitar Amplifier Emulation with Deep Learning," *Applied Sciences*, vol. 10, no. 3, Jan. 2020.

[6] Christian J. Steinmetz and Joshua D. Reiss, "Efficient neural networks for real-time modeling of analog dynamic range compression," in *Audio Engineering Society Convention 152*, May 2022, Audio Engineering Society.

[7] Marco A. Martínez Ramírez, Emmanouil Benetos, and Joshua D. Reiss, "Deep Learning for Black-Box Modeling of Audio Effects," *Applied Sciences*, vol. 10, no. 2, Jan. 2020.

[8] Zhichen Zhang, Edward Olbrych, Joseph Bruchalski, Thomas J. McCormick, and David L. Livingston, "A Vacuum-Tube Guitar Amplifier Model Using Long/Short-Term Memory Networks," in *SoutheastCon 2018*, St. Petersburg, FL, Apr. 2018, IEEE.

[9] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," *arXiv:1609.03499 [cs]*, Sept. 2016.

[10] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang, "A Comprehensive Survey of Neural Architecture Search: Challenges and Solutions," *ACM Computing Surveys*, vol. 54, no. 4, May 2021.

[11] Hadjer Benmeziane, Kaoutar El Maghraoui, Hamza Ouarnoughi, Smail Niar, Martin Wistuba, and Naigang Wang, "A Comprehensive Survey on Hardware-Aware Neural Architecture Search," *arXiv:2101.09336 [cs]*, Jan. 2021.

[12] Michael Stein, "IDMT-SMT-Audio-Effects Dataset," *10.5281/zenodo.7544032*, Jan. 2023.

[13] Udo Zölzer, *Digital Audio Signal Processing*, Wiley, Hoboken, NJ, 3rd edition, 2022.

[14] Jatin Chowdhury, "A Comparison of Virtual Analog Modelling Techniques for Desktop and Embedded Implementations," *arXiv:2009.02833 [cs, eess]*, Sept. 2020.

[15] Michael Farnsworth, Ashutosh Tiwari, and Meiling Zhu, "Multi-level and multi-objective design optimisation of a MEMS bandpass filter," *Applied Soft Computing*, vol. 52, Mar. 2017.

[16] Alec Wright and Vesa Välimäki, "Neural Modelling of Periodically Modulated Time-Varying Effects," in *Proceedings of the 23rd International Conference on Digital Audio Effects*, Vienna, Austria, Sept. 2020.

[17] Alec Wright and Vesa Välimäki, "Grey-Box Modelling of Dynamic Range Compression," in *Proceedings of the 25th International Conference on Digital Audio Effects*, Vienna, Austria, Sept. 2022.

[18] Marco A. Martínez Ramírez, Oliver Wang, Paris Smaragdis, and Nicholas J. Bryan, "Differentiable Signal Processing With Black-Box Audio Effects," in *International Conference on Acoustics, Speech and Signal Processing*, Toronto, ON, Canada, June 2021, IEEE.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, MIT Press, Cambridge, MA, Nov. 2016.

[20] Oliver Nelles, *Nonlinear System Identification: From Classical Approaches to Neural Networks, Fuzzy Models, and Gaussian Processes*, Springer, 2nd edition, 2020.

[21] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, Honolulu, Hawaii, USA, Jan. 2019, AAAI Press.

[22] Christian J. Steinmetz, Jordi Pons, Santiago Pascual, and Joan Serra, "Automatic Multitrack Mixing With A Differentiable Mixing Console Of Neural Audio Effects," in *International Conference on Acoustics, Speech and Signal Processing*, Toronto, ON, Canada, June 2021, IEEE.