Audio Engineering Society

Conference Paper 8

Presented at the AES International Conference on Artificial Intelligence and Machine Learning in Audio
2025 September 8–10, London, UK

Transfer Learning for Neural Modelling of Nonlinear Distortion Effects

Tara Vanhatalo[1,2,3], Pierrick Legrand[1], Myriam Desainte-Catherine[2], Pierre Hanna[2], Guillaume Pille[3], Antoine Brusco[3], and Joshua D. Reiss[4]

[1]*Centre Inria de l'Université de Bordeaux, Institute of Mathematics of Bordeaux, UMR 5251 CNRS, University of Bordeaux, France*
[2]*University of Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, France*
[3]*Orosys SAS, Saint-Gély-du-Fesc, France*
[4]*Centre for Digital Music, Queen Mary University of London, UK*

Correspondence should be addressed to Tara Vanhatalo (`tara.vanhatalo@inria.fr`)

ABSTRACT

Modelling analogue distortion replicates the warmth, character, and nonlinear harmonics of classic gear often used in audio production, to be incorporated into modern, digital workflows. Creating such models can be challenging as it requires accurately capturing complex nonlinearities and dynamic responses while while ensuring real-time processing. We evaluate the performance of various transfer learning methods for improved modelling of distortion effects. These approaches take into account the choice of pretrained model, the choice of dataset for fine-tuning, and the method in which the knowledge is transferred from source to target task. We compare the impact of using a source effect closest to the target one for pretraining, determined either via feature extraction or using domain knowledge, to the use of a "general model". We evaluate how to transfer the knowledge from source to target task by assessing which weights to freeze and what data to use for the downstream training. Each approach is assessed on two architectures applied to three example distortion modelling tasks. All models obtained via transfer learning are compared to a baseline model trained from scratch with random initialisation to gauge the gain in accuracy due to the transfer. Results show that transfer learning can help improve model performance and reduce training times when the appropriate transfer approach is used.

1 Introduction

Many iconic audio devices, such as tube amplifiers, tape machines, and vintage pedals, have unique distortion characteristics that are highly sought after. Modelling these effects allows musicians and producers to access these sounds without needing expensive or hard-to-find hardware, and allows for them to be used within digital workflows seamlessly.

One approach to modelling audio distortion effects is to use neural networks in a black-box training pipeline. The training data used to create these models consists of audio signals - such as instrument recordings or synthetic test signals - sent through the target analogue devices, the output of which is recorded and used as ground truth. Reducing the amount of training data and time has a number of benefits. These include: a reduced risk of error in the data generation portion that can allow for non-specialists to create their own models; less wear-and-tear of the equipment used to create the training data; and reduced environmental impact.

Transfer Learning (TL) is a commonly used method in machine learning, most often applied to cases where there is insufficient training data. It transfers the knowledge from one or more source tasks to a target task and can manifest in a variety of methods. We give a formal definition of transfer learning, adapted from Pan and Yang [1]. In our case there is only one source task but this can easily be extended to the general case.

Given a source domain $\mathbf{D}_S$ and task $\mathbf{T}_S$, a target domain $\mathbf{D}_T$ and task $\mathbf{T}_T$, transfer learning aims to improve the learning of the target predictive function $f(.)$ in $\mathbf{D}_T$ using the knowledge in $\mathbf{D}_S$ and $\mathbf{T}_S$ where $\mathbf{D}_S \neq \mathbf{D}_T$ or $\mathbf{T}_S \neq \mathbf{T}_T$. Here a domain is defined as $\mathbf{D} = \{\mathbf{X}, P(X)\}$ where $\mathbf{X}$ is a feature space and $P(X)$ is a marginal probability distribution, and a task is defined as $\mathbf{T} = \{\mathbf{Y}, f(.)\}$ where $\mathbf{Y}$ is a label space and $f(.)$ is a predictive function learned from training data. In the case of Deep Transfer Learning (DTL), this predictive function takes the form of a neural network [2], [3].

DTL started to arise when Deep Neural Networks (DNN) were used as feature extractors in TL models, a process referred to as "incomplete" transfer in Yu et al. [3]. Around the same time as the incorporation of DNNs into traditional TL, model-based DTL methods also started to emerge [3]. Model-based approaches consist of sharing and fine-tuning parameters of models. In this work, we focus on transfer learning via fine-tuning of the model weights for audio effects modelling.

Little work has explored the possibility of applying TL to neural networks for audio effects modelling. The neighbouring field of Music Information Retrieval (MIR) contains a number example works using TL for various tasks. In Hamel et al. [4], TL applied to genre classification, automatic tag annotating and music similarity is investigated. Oord et al. [5] exploit an available large-scale music dataset, the Million Song Dataset (MSD), for classification tasks on other datasets by reusing models trained on MSD for feature extraction. In Liang et al. [6], a multi-layer neural network trained for semantic tagging prediction is used as a content model for content-aware music recommendation. In Choi et al. [7], a tagging CNN trained on MSD is used to create explicit labels for the downstream task of playlist generation using RNN. In Choi et al. [8], TL takes the form of a pretrained feature extraction network for both music classification and regression tasks. Alfaro-Contreras et al. [9] examine the possible improvements owed to TL using either image or audio data for Optical Music Recognition and Automatic Music Transcription.

In this work, we apply model-based DTL to improve the performance of state-of-the-art static models for audio effects modelling. Although transfer learning is often used in contexts where the downstream (i.e. target) task lacks sufficient labelled data, it can also lead to faster learning, more accurate predictions and/or require less training data. This is of interest for virtual analogue modelling as we could, in theory, train one model on a large amount of data acquire from a chosen audio effect, and then use this trained network as a starting point for all other models of a variety of different, even unrelated, black-box audio effects we wish to create.

In this case the source task is training a model of a chosen audio effect S and the target task is training a model of a different audio effect T. The initial, large training dataset could even be synthetic, using circuit simulation techniques to create the data to avoid the wear-and-tear of electrical components in analogue devices and to minimise the risk of human error during the capture process. Thus, utilising transfer learning could allow for reduced training times in the target tasks which would incur shorter times to create new models and reduce the required computational resources, and thus environmental impact.

These benefits are amplified in the case of parametric models. When we wish to model the behaviour of the user controls' influence on the output audio and have the simulation adapt to different control settings, the amount of training data required increases exponentially when sampling combinatorically which is often the method employed [10]. Reduced amounts of necessary training data are therefore doubly important in such cases.

We therefore evaluate the impact on overall performance of applying transfer learning to this modelling task. We also determine the most effective methods of conducting the pretraining and fine-tuning phases of the knowledge transfer. A case study is carried out on three example distortion effects with increasing degrees of nonlinearity: an overdrive, a distortion and a fuzz which we aim to model with two example networks, a simple LSTM recurrent network and the feedforward WaveNet, from Wright et al. [11].

The organisation of this paper is as follows. In Section 2, we present the methodology behind both the pretraining and fine-tuning experiments. In Section 3, we present the experimental results and comparison and discuss the findings. All the data used in this work is made available at: `https://www.math.u-bordeaux.fr/~plegra100p/NN/TL`.

2 Methodology

For the pretraining, illustrated at the top right of Figure 1, we compare which type of starting model produces the highest accuracy between a model most similar to the target device and a general-purpose model that can be used regardless of what the target device is. In order to determine the similarity between devices, we use a Convolutional Neural Network (CNN) pretrained for distortion effects classification, dubbed FxNet, to extract features from a number of devices [12]. The closest model is chosen as the one whose features minimise the L_2-distance with the features of a given target effect. Similarly, the general-purpose model chosen is the one whose features have the smallest distance from an average feature vector computed from all the effects in the dataset. More detail on this is given in Subsection 2.1. In order to ascertain how effective this approach is, we compare results with those obtained using closest and general-purpose models obtained with domain knowledge.

For the fine-tuning phase, illustrated at the bottom right of Figure 1, we evaluate the impact of the type of signal used for the training, the length of this signal (in order to determine a minimum amount of data required), and which weights to freeze for optimal results with both networks. All results obtained after fine-tuning are compared to a baseline model trained from scratch using random initialisation.

2.1 Pretraining

Pretraining-based TL describes the case where a model initially trained for a particular source task with a given data distribution, is adapted to a related task with a different training data distribution. This method generally leads to improved model performance and has the benefit of requiring less data for the target model fine-tuning.

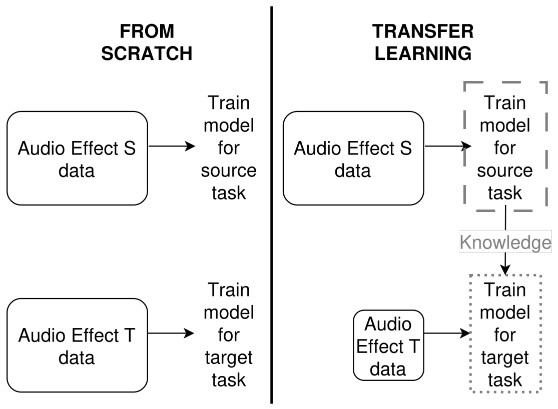

Fig. 1: *Training from scratch vs transfer learning. The pretraining is circled by the dashed line and the finettuning by the dotted line.*

Here, we evaluate the impact on performance that the choice of pretrained source model can have. For this, comparisons are carried out between a source effect that is closely related to the target effect and a source effect that is more generally applicable to various target effects. We call these two choices of pretrained model *closest model* and *general model*.

2.1.1 Choice of Pretrained Model

The transfer of knowledge from source to target task is expected to improve modelling performance. Regarding the choice of pretrained source model, we compare fine-tuning a model deemed closest to the target one with a single model chosen as the starting point for all target model training. This is done in order to gauge whether it is preferable to have very closely related source and target tasks or whether a generally applicable source model is sufficient for the fine-tuning of all effects.

For both experiments, the choice of what constitutes the closest model or the most general model is performed with two methods: one is a choice informed by domain knowledge and another is chosen via feature extraction.

We first present the approach based on domain knowledge and then present the feature extraction in the following subsection. For the closest model, we chose an emulation of the same pedal as the target one but from a different designer: Knowledge is transferred from the MultiDrive Pedal Pro (MPP) TS808 to the Nembrini Audio (NA) TS808, from the MPP RAT to the NA RAT, and from the Distorque Face Bender to the MPP Fuzz Face. Table 1 describes all plugins used and the devices they are based on.

Table 1: *Plugins used for all data generation.*

Designer	Plugin	Emulation of
Audified	MultiDrive Pedal Pro	Ibanez TS808
		Ibanez TS9
		Boss BD2
		Boss OD1
		Boss SD1
		Boss DS1
		Boss MT2
		ProCo RAT
		MXR Distortion +
		Arbiter Fuzz Face
		E-H Big Muff
Mercuriall	Greed Smasher	M-B Grid Slammer
	Metal Area	Boss MT2
	TSC 808 Core	TS808
Analog Obsession	Pig Pie	E-H Russian Big Muff
	Zupaa	Vox Tone Bender
Nembrini Audio	BLACK	ProCo RAT
	Clon Minotaur	Klon Centaur
	808 Overdrive	Ibanez TS808
Distorque Audio	Face Bender	Arbiter Fuzz Face
		Tone Bender MKII

These starting models are compared to those obtained via feature extraction: The closest model to the NA TS808 is the MPP OD1, the closest model to the NA RAT is the NA Klon Centaur, the closest model to the MPP Fuzz Face is the MPP MT2. For the general-purpose pretrained model selected using domain knowledge, we chose the NA Klon Centaur as it is known for its tonal transparency that does not colour the signal [13]. The general model obtained via feature extraction is the MPP MXR distortion emulation.

2.1.2 Feature Extraction

We utilise another form of transfer learning for the feature extraction method presented here. A pretrained CNN designed for distortion effects recognition is used to extract features of the training data for the source model choice. This approach bears similarities to other

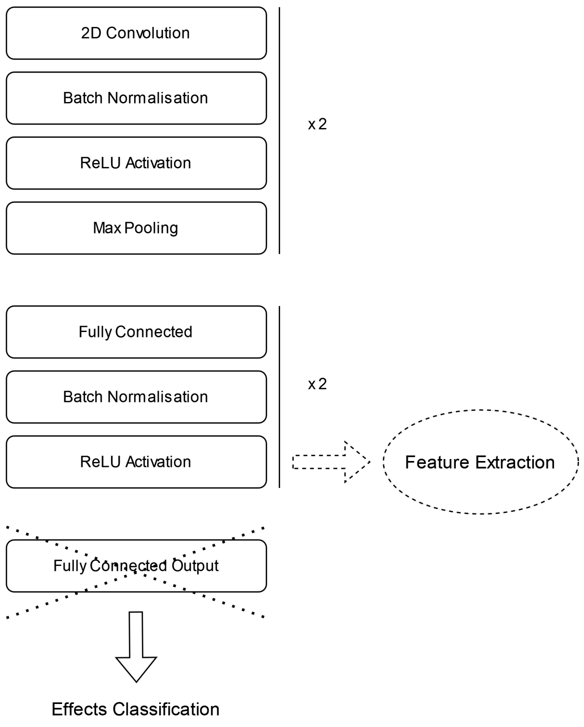

Fig. 2: *The FxNet architecture from [12] adapted for feature extraction.*

such methods employed in MIR, examples of which were presented in Section 1.

The pretrained CNN used in our experiments is the FxNet which comprises two convolutional and three fully connected layers, with batch normalisation at each level. The FxNet model is trained specifically on a number of distortion effects for their classification. We adapt FxNet by removing the output classification layer, which consists of a fully connected layer used to map the learnt feature representation to a vector whose size equals the number of effects in their dataset, as illustrated in Figure 2. The details of the adapted FxNet architecture are presented in Table 2. FxNet takes as input mel power-spectrograms. We extracted these spectrograms from various parts of the recorded signals in our dataset using the Librosa library [14].

We now detail the methods used to determine the closest and general models. We have a full training dataset S_i for each audio effect $i \in N$ listed in Table 1, where N is the total number of audio effects. We denote $\mathbb{D}$ the collection of each of these source datasets $S_i, i \in N$.

Table 2: *Feature Extraction CNN (FxNet) adapted from [12]. #Fmaps is number of feature maps output from the convolutional layers.*

Layer	Size	#Fmaps	Activation
2D Convolution	5×5	6	Linear
Batch Norm.	···	···	···
Activation	···	···	ReLU
Max Pooling	2×2	···	···
2D Convolution	5×5	12	Linear
Batch Norm.	···	···	···
Activation	···	···	ReLU
Max Pooling	2×2	···	···
Fully Connected	120	···	Linear
Batch Norm.	···	···	···
Activation	···	···	ReLU
Fully Connected	60	···	Linear
Batch Norm.	···	···	···
Activation	···	···	ReLU

We apply the modified FxNet model to each source dataset to extract the features $F(S_i)$ of each dataset S_i, as illustrated in Figure 2. We then choose the i^{th} audio effect to be our target effect, and the S_j providing the smallest distance as the source data for the pretraining of the closest model:

$$\underset{F(S_j)}{\arg\min} ||F(S_i) - F(S_j)||_2, \quad j \neq i \tag{1}$$

The target dataset T_i corresponding to the i^{th} audio effect is chosen depending on the fine-tuning method.

Similarly, for the general model the same feature extraction process is applied to obtain all the features of each dataset in $\mathbb{D}$. We then compute an average feature vector:

$$F(A) := \frac{1}{N}\sum_{j=1}^{N} F(S_j) \tag{2}$$

For each $S_j \in \mathbb{D}$, we then compute the Euclidean distance between each potential source dataset features and the average feature vector. The closest feature vector $F(S_j)$ to the average $F(A)$ gives us the source data S_j used to train the general model.

$$\underset{F(S_j)}{\arg\min} ||F(A) - F(S_j)||_2 \tag{3}$$

Again, the target dataset T_i corresponding to the i^{th} audio effect is chosen depending on the fine-tuning method, a number of which are also evaluated in this work.

2.2 Fine-Tuning

In the fine-tuning phase, we adapt the pretrained models to a new target effect by retraining on the target task data and optionally freezing certain network parameters.

2.2.1 Choice of Dataset

For this study, we assembled a dataset similar to the one proposed by Comunità et al. [12]. The input data comprises samples from the IDMT-SMT-Guitar dataset [15] and a number of artificial test signals including Dirac delta functions, step functions, exponential sine sweeps and sawtooth waveforms of varying amplitude. The guitar signals were all taken from the IDMT-SMT-Guitar dataset 4 and vary in genres and tempos. These signals were then processed using the same plugins as those used in Comunità et al. [12] and three additional distortion effect plugins, all of which are summarised in Table 1. All pretrained models were obtained using the whole dataset whereas the type and length of the data used for fine-tuning was varied in order to gauge its impact on target task performance.

We evaluate the impact of the type of training data employed for the fine-tuning by comparing the use of the artificial test signals to the recorded guitar and by comparing input data seen during pretraining to unseen data. All the data used for this comparison is of same length in order to isolate the impact of the signal content. We also study the impact of signal length on the fine-tuning accuracy in order to gauge a minimal amount of data necessary for this phase. Only the recorded guitar was used for this experiment, with length varying from five seconds to nine minutes.

2.2.2 Weight Freezing

It is common practice in transfer learning approaches to freeze certain weights of the pretrained model for the fine-tuning. We study the impact of weight freezing for the transfer by varying the layers whose weights are fixed in our example networks.

The recurrent model is fairly shallow and contains only two layers that can be frozen. Therefore we compare the impact of freezing the recurrent layer's weights, the fully connected layer's weights, and complete retraining. Yosinksi et al. observe experimentally that DNNs appear to exhibit more general features that are more applicable to many datasets and tasks in their initial layers whereas more specific features tend to appear in the later layers [16]. Therefore for the example convolutional model we have chosen, we compare freezing the initial block to freezing all convolutional layers and to complete retraining.

3 Results

We now present the results of a case study using two illustrative architectures for audio effects modelling: A simple single-layer LSTM network with a fully connected output and 32 recurrent units; and the feedforward WaveNet from Wright et al. [11] with two blocks of ten convolutional layers with a dilation factor of two. Each convolutional layer has 16 filters and a kernel size of three. We evaluate various methods of TL using these two models on three different distortion effects using a variety of methods in order to encourage significance of the results presented here.

3.1 Choice of Pretrained Model

All pretraining was carried out using the entire dataset which comprises around ten minutes of guitar recordings and a number of artificial test signals as described previously. This dataset was split, using 90% for training and 10% for validation. Both the recurrent and convolutional models were trained using the same optimisation and data hyperparameters for a total of 50 epochs. The models are then fine-tuned using the same hyperparameters for another 50 epochs. In the case of the training from scratch, all models are trained on the fine-tuning data for 100 epochs. Mean Absolute Error (MAE) was used as the loss function for all training, as well as the Adam optimiser with an initial learning rate of 0.005 and weight decay of 1e-4. A batch size of 50 was used for the recurrent model and 40 was used for the convolutional one. All errors reported are also expressed in terms of MAE after training the models for 100 epochs and are presented in Table 3. All LSTM models showed an improvement when using transfer learning however the WaveNet models did not always present an improvement. In most cases, the general-purpose pretrained model outperforms other methods, with the feature extraction-based general-purpose model showing the lowest error most consistently.

3.2 Weight Freezing

In Table 4 and Table 5, we report the average test losses of the different weight freezing approaches again after 100 epochs training.

Freezing the weights of the recurrent layer of the LSTM model for fine-tuning leads to a significant increase in error. Freezing the fully connected layer of this model can lead to an improvement in accuracy. However, for the distortion emulation this freezing method leads to an overall decrease in performance. Retraining the models consistently leads to a decrease in test MAE for the LSTM model. Transfer learning rarely leads to decreased error for the convolutional models when compared to training from random initialisation for the same number of epochs. This could be due to the dataset size as increased labelled data or the use of data augmentation at fine-tuning has been found to reduce the effects of the knowledge transfer in certain convolutional architectures [17].

3.3 Subjective Evaluation

To better evaluate the impact of transfer learning on model performance for equal training length, we implement subjective listening tests using the Multiple Stimuli with Hidden Reference and Anchor (MUSHRA) methodology with the webMUSHRA framework [18]. In these listening tests, participants are asked to rate, out of 100, the similarity of audio extracts compared to a reference. The audio clips evaluated contain both simulations obtained with and without transfer learning, a hidden reference and two anchors. The MUSHRA methodology allows for statistical significance of results with fewer participants [19]. This is to assess whether the slight decreases in error observed with the best performing models in Tables 4 and 5 are audibly significant. A total of 40 assessors were used for the subjective evaluation. These assessors are a mix of audio professionals with experience in critical listening and guitarists.

The results are presented in Figures 3 and 4. Certain outliers were omitted from these results if the assessor assigned a score lower than 90 for the reference or higher than 50 for either of the anchors. For both the LSTM and WaveNet models, a relatively high variance in responses can be observed. This is most likely due to the fact that the figures show an aggregate of responses, evaluating all effects types including the fuzz effects

Table 3: *Final model MAE with respect to the choice of pretrained model for each class of distortion effect. Pretrained model chosen either using Domain Knowledge (DK) or via Feature Extraction (FE).*

	Overdrive		Distortion		Fuzz	
	LSTM	WaveNet	LSTM	WaveNet	LSTM	WaveNet
Closest Model *(DK)*	0.0307	0.0281	0.0381	0.0193	0.0744	0.0557
Closest Model *(FE)*	0.0297	0.0288	0.0260	0.0268	0.0817	**0.0518**
General Model *(DK)*	**0.0280**	0.0307	0.0260	0.0268	0.0719	0.0576
General Model *(FE)*	0.0290	**0.0267**	**0.0135**	0.0162	**0.0652**	0.0651
From Scratch	0.0308	0.0295	0.0278	**0.0146**	0.0902	0.0530

Table 4: *Average weight freezing MAE (LSTM).*

LSTM			
	Overdrive	Distortion	Fuzz
Freeze rec. layer	0.0512	0.2693	0.1765
Freeze dense layer	**0.0290**	0.0313	**0.0724**
Retrain	0.0293	**0.0259**	0.0773
From scratch	0.0308	0.0278	0.0902

which are known to be difficult to model with these approaches [20].

A Student's t-test was used over ANOVA to compare the means of the groups as there are only two present here. A one-sided test was used because we assume there will be an improvement in performance using TL for the alternative hypothesis. Regarding the LSTM model, the mean score of the TL simulations is 16% higher than that of the from scratch results with a p-value of 0.003 computed with Student's t-test from the SciPy Python library [21]. The listening tests back up the objective measures that showed improvements in simulation accuracy when using TL on LSTM. The lack of improvement seen in the WaveNet predictions is also reflected in the listening test results. The mean score sees a decrease of 2.4% when using TL but the p-value computed equals 0.57 and therefore this result was not deemed statistically significant. The listening tests back up the objective measures that showed no improvements in accuracy when using TL on the WaveNet. The interquartile range for both simulations from TL and trained from scratch remains about the same, with the median seeing a 10% decrease with TL.

Table 5: *Average weight freezing MAE (WaveNet).*

WaveNet			
	Overdrive	Distortion	Fuzz
Freeze conv. layers	0.0396	0.1734	0.1558
Freeze first block	0.0311	0.0328	0.0887
Retrain	**0.0286**	0.0223	0.0581
From scratch	0.0295	**0.0146**	**0.0530**

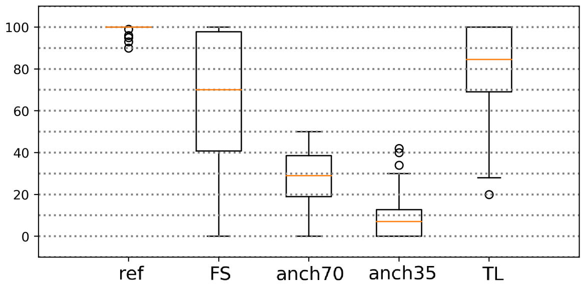

Fig. 3: *MUSHRA results for the LSTM model. The scores from the reference (ref), simulations obtained using Transfer Learning (TL) and training From Scratch (FS) as well as from the hidden anchors are shown.*

3.4 Fine-Tuning Data

We then evaluate what data to use for the fine-tuning phase, both in terms of signal content and dataset length. All results presented hereafter were obtained when fine-tuning the general-purpose model chosen via feature extraction without weight freezing as this provided the most consistent improvements in the previous experiments. In Table 6, we compare two subsets containing recorded guitar with a dataset containing exclusively artificial test signals. These results were obtained by retraining the models during fine-tuning as this is the method that performed best for both example models in the previous experiments. In almost all instances, except the distortion emulation using the WaveNet model, the guitar datasets outperformed the use of test signals. Moreover, in all

Table 6: *Fine-tuning dataset type. All datasets are of equal length (37 seconds).*

	Overdrive		Distortion		Fuzz	
Signal Type	LSTM	WaveNet	LSTM	WaveNet	LSTM	WaveNet
Recorded Guitar *(seen at pretraining)*	0.0452	0.0301	0.0758	0.1015	**0.0886**	0.0853
Test Signals *(seen at pretraining)*	0.0656	0.0655	0.2202	**0.0480**	0.1133	0.1466
Recorded Guitar *(unseen at pretraining)*	**0.0305**	**0.0290**	**0.0344**	0.0583	0.0958	**0.0772**

Table 7: *Fine-tuning dataset length. The from scratch results were trained on the full guitar dataset, lasting 546 seconds. Training times for LSTM recorded on an Nvidia GeForce GTX 1650 GPU, training times for WaveNet recorded on an Nvidia GeForce RTX 2060 GPU.*

	Overdrive		**Distortion**		**Fuzz**		**Training Time(s)**	
Signal Length (s)	LSTM	WaveNet	LSTM	WaveNet	LSTM	WaveNet	LSTM	WaveNet
5	0.0577	0.0534	0.1451	0.0762	0.1457	0.1369	0.51	3.66
30	0.0359	0.0299	0.0767	0.1081	0.1378	0.1098	2.94	19.97
210	**0.0286**	0.0252	0.0355	0.0328	0.0803	0.0788	21.09	190.89
540	0.0301	**0.0225**	**0.0126**	0.0264	**0.0582**	0.0559	42.83	396.85
From Scratch	0.0308	0.0295	0.0278	**0.0146**	0.0902	**0.0530**	57.80	381.28

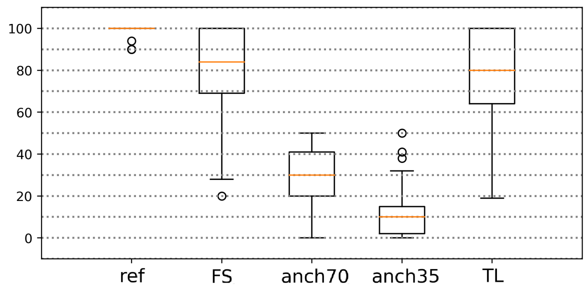

Fig. 4: *MUSHRA results for the WaveNet model. The scores from the reference (ref), simulations obtained using Transfer Learning (TL) and training From Scratch (FS) as well as from the hidden anchors are shown.*

cases except one, training on unseen data outperformed training on a subset of the source data.

Our final experiment aims to find a trade off between training time and model accuracy. All models obtained via transfer learning were fine-tuned on recorded guitar signals of various lengths. We compare these to from scratch training on the full guitar subset in Table 7.

For the simpler emulation task of the overdrive effect, we obtain comparable error for both the recurrent and convolutional models when using transfer learning on a significantly reduced dataset. We go from full training on nine minutes and six seconds of data to 30 seconds resulting in a decrease in training time by around 95% for both the LSTM model and WaveNet for a slight increase in MAE. For more complex effects, the increase in MAE is more pronounced. It is still possible to cut training time in half for a maximum increase in error of 2.6%. Moreover, transfer learning for the fuzz emulation using LSTM leads to a 1% decrease in error while reducing training time by over 50%.

4 Conclusion

In this paper we studied the impact of transfer learning and its implementation for distortion effects modelling. We evaluated this approach on three example modelling tasks, using two illustrative networks. Transfer learning shows a slight decrease in error for the recurrent model compared to training from scratch for an equal number of epochs. However, the convolutional network did not show any significant improvements. This was also reflected in the subjective results from listening tests. Using a general-purpose pretrained model obtained via feature extraction presents the most consistent improvements in performance. It was found that fine-tuning on guitar recordings unseen during pretraining lead to the best performance and significant model training speedups can be achieved using for minimal to no increase in error.

Appropriate weight freezing should be employed in order to avoid any increase in error. Namely, for consistent improvements, complete retraining is preferred. In order to increase the statistical significance of these results, it would be of interest to rerun these experiments a number of times while varying the random seed. This case study could also be extended to other audio effects, especially those presenting different audio behaviour as we have constrained ourselves to distortion effects in this work.

Finally, extending this work to parametric models that incorporate user-facing controls is of great interest as the benefits of transfer learning are especially prominent in this case.

5 Acknowledgments

This research was hosted by the SCRIME of the University of Bordeaux (Studio of Creation and Research in Computer Science and Experimental Music) which is funded by the french ministry of culture. Experiments presented in this paper were carried out using the PlaFRIM experimental testbed, supported by Inria, CNRS (LABRI and IMB), Université de Bordeaux, Bordeaux INP and Conseil Régional d'Aquitaine (see `https://www.plafrim.fr`).

References

[1] Pan, S. J. and Yang, Q., "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, 22, pp. 1345–1359, 2010, ISSN 10414347, doi:10.1109/TKDE.2009.191.

[2] Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., and Liu, C., "A Survey on Deep Transfer Learning," in *Artificial Neural Networks and Machine Learning – ICANN*, 2018.

[3] Yu, F., Xiu, X., and Li, Y., "A Survey on Deep Transfer Learning and Beyond," *Mathematics*, 10, 2022, ISSN 22277390, doi:10.3390/math10193619.

[4] Hamel, P., Davies, M. E. P., Yoshii, K., and Goto, M., "Transfer Learning in MIR: Sharing Learned Latent Representations for Music Audio Classification and Similarity," in *Proceedings of the 14th International Society for Music Information Retrieval Conference*, pp. 9–14, 2013.

[5] Oord, A. V. D., Dieleman, S., and Schrauwen, B., "Transfer Learning by Supervised Pre-Training for Audio-Based Music Classification," in *Proceedings of the 15th International Society for Music Information Retrieval Conference*, pp. 29–34, 2014.

[6] Liang, D., Zhan, M., and Ellis, D. P. W., "Content-Aware Collaborative Music Recommendation Using Pre-Trained Neural Networks," in *Proceedings of the 16th International Society for Music Information Retrieval Conference*, pp. 295–301, 2015.

[7] Choi, K., Fazekas, G., and Sandler, M., "Towards Playlist Generation Algorithms Using RNNs Trained on Within-Track Transitions," in *Workshop on Surprise, Opposition, and Obstruction in Adaptive and Personalized Systems*, 2016, doi:10.1145/1235.

[8] Choi, K., Fazekas, G., Sandler, M., and Cho, K., "Transfer learning for music classification and regression tasks," in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, pp. 141–149, 2017.

[9] Alfaro-Contreras, M., Valero-Mas, J. J., Iñesta, J. M., and Calvo-Zaragoza, J., "Insights Into Transfer Learning Between Image and Audio Music Transcription," in *19th Sound and Music Computing Conference*, pp. 295–301, 2022.

[10] Juvela, L., Damskägg, E.-P., Peussa, A., Mäkinen, J., Sherson, T., Mimilakis, S. I., Rauhanen, K., and Gotsopoulos, A., "End-to-End Amp Modeling: from Data to Controllable Guitar Amplifier Models," in *2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, Institute of Electrical and Electronics Engineers, 2023, doi:10.1109/icassp49357.2023.10094769.

[11] Wright, A., Damskägg, E. P., Juvela, L., and Välimäki, V., "Real-time Guitar Amplifier Emulation with Deep Learning," *Applied Sciences (Switzerland)*, 10(3), 2020, ISSN 20763417, doi:10.3390/app10030766.

[12] Comunità, M., Stowell, D., and Reiss, J. D., "Guitar Effects Recognition and Parameter Estimation With Convolutional Neural Networks," *Journal of*

the Audio Engineering Society, 69, pp. 594–604, 2021, ISSN 15494950, doi:10.17743/jaes.2021.0019.

[13] World, G., *Guitar World Presents 200 Stompbox Reviews*, Hal Leonard, 2014, https://www.halleonard.com/product/123825/iguitar-worldi-presents-200-stompbox-reviews.

[14] McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., and Nieto, O., "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, volume 8, 2015, https://librosa.org/.

[15] Kehling, C., Männchen, A., and Eppler, A., "IDMT-SMT-Guitar," Technical report, Fraunhofer Institute for Digital Media Technology IDMT, 2014, https://www.idmt.fraunhofer.de/en/publications/datasets/guitar.html.

[16] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H., "How transferable are features in deep neural networks?" in *Advances in Neural Information Processing Systems 27 (NIPS '14)*, 2014.

[17] Zoph, B., Ghiasi, G., Lin, T.-Y., Cui, Y., Liu, H., Cubuk, E. D., and Le, Q. V., "Rethinking Pre-training and Self-training," in *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020.

[18] Schoeffler, M., Bartoschek, S., Stöter, F.-r., Roess, M., Edler, B., and Herre, J., "webMUSHRA — A Comprehensive Framework for Web-based Listening Tests," *Journal of Open Research Software*, 6, p. 8, 2018.

[19] ITU-R, "ITU-R BS.1534-3: Method for the Subjective Assessment of Intermediate Quality Level of Audio Systems," *Intern. Telecom. Union Radiocom. Sector*, BS Series, p. 34, 2015.

[20] Comunità, M., Steinmetz, C., Phan, H., and Reiss, J., "Modelling Black-Box Audio Effects with Time-Varying Feature Modulation," pp. 1–5, 2023, doi:10.1109/ICASSP49357.2023.10097173.

[21] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, 17, pp. 261–272, 2020, doi:10.1038/s41592-019-0686-2.