Using quality and intelligibility measures to create an estimator for reverberation time in a shoebox-shaped room with a multilayer perceptron model

Anneline Lucia Mol[1]

Supervisor(s): Jorge Martinez Castaneda, Dimme De Groot

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
July 8, 2024

Name of the student: Anneline Lucia Mol
Final project course: CSE3000 Research Project
Thesis committee: Jorge Martinez Castaneda, Dimme de Groot, Sole Pera

Using quality and intelligibility measures to create an estimator for reverberation time in a shoebox-shaped room with a multilayer perceptron model

Anneline Lucia Mol, *Supervisors:* Jorge Martinez Castaneda, Dimme de Groot

***Abstract*—Reverberation is a key aspect when designing the interior of buildings, and must be carefully considered in the context of the function of the room. Defined by the reverberation time (RT), it is known to have a big influence on the intelligibility and quality of audio in closed spaces. In this work, we investigate the relationship between the RT and explore the feasibility of using multilayer perceptron (MLP) networks to create an estimator for the RT by using the values of objective measures as input features. We investigate five measures in particular: the Perceptual Evaluation of Speech Quality (PESQ), Virtual Speech Quality Objective Listener (ViSQOL) and its extension focused on audio (ViSQOLAudio), and the Short-time Objective Intelligibility Measure (STOI) and its extension ESTOI. We create a 3-layer MLP network that estimates the RT with a mean absolute error of 0.144 on our simulated RIR test sets and 0.196 on our real RIR test set.**

***Index Terms*—Audio Quality, ESTOI, Multilayer Perceptron, Objective Measures, PESQ, Reverberation Time, Speech Intelligibility, STOI, ViSQOL, ViSQOLAudio**

I. Introduction

From at home settings with loudspeaker placement to more professional settings such as indoor architecture, reverberation is a major aspect in how sound is perceived within an enclosed space, and is therefore of great interest [1], [2]. It is often defined in terms of the Reverberation Time (RT), described using the T_{60} metric. This is the time taken in seconds for the amplitude of a room impulse response (RIR), the characterising sound of an enclosing space, to reduce by 60 dB, the level at which the sound falls outside the perceptual range of hearing [3].

There are currently multiple ways to obtain a measurement of the RIR in a room and extract the RT [4], [5], [6]. However, the cost and requirements of these methods leads to question whether or not it may be possible to estimate the RT reliably and with reasonable accuracy with a less resource-intensive method instead.

Currently, much work has been done to estimate the RT blindly using machine learning, where the use of a convolutional neural network (CNN) with audio as input is particularly popular. These attain accuracies with the Mean Squared Error (MSE) of predictions ranging from 0.03 to 0.07 when tested on real RIRs [7], [8]. However, they also have complex models with little insight to their workings due to the high number of input parameters and are therefore often computationally expensive [9]. This warrants the question whether a simpler model can be created.

Audio quality and speech intelligibility are two concepts that have been shown to be quite dependent on reverberation time [2]. Speech intelligibility refers to how well words present in audio can be identified and understood while perceptual audio quality, refers to how sound is experienced by a human listener. Both concepts are mostly rooted in subjective experience and evaluation with methods such as MUSHRA, but objective measures have been developed to replicate expected subjective quality or intelligibility scores and have been evaluated and compared [10], [11], [12]. Xia et al. [13] have shown that there is a negative relation between reverberation and intelligibility. Similarly, Ratnam et al. [14] express that the RT is of importance to both audio quality and speech intelligibility.

In this research, we are interested to see if there is a relationship between the outputs of these measures and the RT value and whether the outputs of these measures can serve as features to form an estimator for the RT using a multi-layer perceptron (MLP) model, which is a simple form of artificial neural network (ANN). This unconventional yet innovative take on using quality and intelligibility measures can help create a model that could be a lot simpler than the earlier mentioned neural networks that take audio as an input, allowing for easier insight into the role of the parameters and a lower running cost. This leads to the following main research question:

> Can objective audio quality and speech intelligibility measures be used to estimate the reverberation time through the use of a multilayer perceptron network with performance comparable to other state-of-the-art estimators?

This paper explores the dependencies between different audio quality and intelligibility measures and the reverberation occurring in the acoustic scene, investigating the feasibility of creating an estimator for reverberation time from these quality measures. The set of measures evaluated contains the following: Perceptual Evaluation of Speech Quality (PESQ) [15], Virtual Speech Quality Objective Listener (ViSQOL) [16], its audio equivalent ViSQOLAudio [17], and Short-time Objective Intelligibility measure (STOI) [18] and its extension Extended Short-time Objective Intelligibility measure (ESTOI)

[19]. These are individually experimented upon before being combined to create a new estimator for reverberation time using a MLP network. The resulting model is then evaluated.

Section II elaborates on the measures considered in this project. Section III then outlines the methodology followed during this project after which Section IV explains particular choices made in the research to ensure our research was responsible. Section V details the results obtained from the experiment whereafter Section VI concludes and provides ideas for potential future investigation.

II. Objective Measures

Though many objective quality and intelligibility measures exist, this research focuses on using only a selected subset of objective and intrusive measures. These measures require both the clean reference audio and degraded audio to calculate a score. Below each of the selected measures is briefly elaborated upon and the motivation for use is given. Lastly, some measures not selected are explained.

A. Perceptual Evaluation of Speech Quality (PESQ)

PESQ is a standard objective speech-focused quality measure that was developed for predicting what the subjective quality of the input would be when looking at telephony [15]. The algorithm functions by first identifying delays between the original and degraded input, and then using this delay set to compare the original and degraded audio using an internal perceptual model. The key to this for PESQ is that the internal representation of the signals is 'analogous to the psychophysical representation of audio signals in the human auditory system' [20]. From this, a Mean Opinion Score (MOS) score is calculated.

Although this recommendation is no longer in force and replaced by recommendation P.863, Perceptual Objective Listening Quality Assessment (POLQA), PESQ as a measure still performs well compared to other quality measure (QM)s, motivating us to use this measure [11]. Additionally, research has shown that PESQ has a decent correlation to reverberation features such as the reverberation tail effect [21]. We use an open Python implementation available from Wang et al. and run the samples on the wide-band version of the measure with a sampling rate of 16 kHz [22]. As our audio samples have a sample rate of 48 kHz, these are resampled before being measured by the algorithm.

B. Virtual Speech Quality Objective Listener (ViSQOL)

ViSQOL is another intrusive QM which was made to model human sensitivity to degradations in speech quality [16]. Being developed with the weaknesses of PESQ such as clock drift in mind, a bigger focus is placed on which types of degradation are noticeable to the human ear. The algorithm works by identifying corresponding patches of interest and compares the similarities of their spectrograms to calculate a similarity score.

In this project, we use the implementation provided by the Audio Toolbox in MATLAB 2024a [23]. Another implementation openly available is mentioned by Hines et al. [17].

C. Virtual Speech Quality Objective Listener for Audio (ViSQOLAudio)

ViSQOLAudio [17] is an adaptation of ViSQOL that generalises beyond speech to consider the full audio spectrum, becoming a more general system [11]. Rather than isolating patches of sound to compare, ViSQOLAudio considers all patches important. The number of frequency bands evaluated have also been increased to accommodate for a human's full range of hearing, rather than only focusing on bands used in speech. The measure then outputs the result on a similarity scale from 0 to 1.

We choose to include both the original ViSQOL and ViSQOLAudio measure as they are created for different types of audio and may therefore exhibit different behaviour depending on the clean speech provided. Similarly to ViSQOL, we use the implementation provided by MATLAB 2024a's Audio Toolbox.

D. Short-time Objective Intelligibility measure (STOI)

STOI, as opposed to the other measures, is a speech intelligibility metric rather than a quality metric. Created to handle speech degraded using time-frequency techniques [12], the measure first turns both the original and degraded audio into frames in time-frequency representation before calculating an intermediate score and averaging this over all bands and frames to create the final score [18]. We use a publicly available Python implementation based on the original Matlab implementation available from [24].

E. Extended Short-time Objective Intelligibility measure (ESTOI)

ESTOI is another intelligibility metric and takes inspiration from STOI [19]. It expands to work for a larger range of input signals and distinguishes itself from STOI by not assuming mutual independence between frequency bans and incorporating spectral correlation. The measure first extracts the temporal envelopes of the subbands of both the clean and degraded audio before normalising the resulting spectrograms. On these, the "distance" of each band is computed and then averaged to give the final intelligibility index. We use the implementation of ESTOI from the same library as that of STOI.

F. Other measures considered

Beside the measures chosen explained above, several other measures have been considered for use. Table I lists the other metrics considered with reasons for exclusion.

III. Methodology

Using the reasoning mentioned in Section II to establish a set of measures to investigate, the experiment can be divided into four parts: creating the dataset of clean and degraded audio, generating the feature dataset, creating and training the MLP, and testing the MLP. These individual parts are explained in subsections III-A, III-B, and III-C. Figure 1 shows a graphical representation of the pipeline that a pair of clean

Measure	Noise Robustness
PESQ	Previous ITU standard and wide use, previously used in the context of reverberant audio [21], [25].
PEASS	Very high computation time [26].
ViSQOL	Focus on human-perceptible degradation and open-source measure.
ViSQOL-Audio	Purpose to rate quality of music and audio samples may react to the RT differently than other measures.
POLQA	No non-commercial licences of the software available.
PEAQ	Non-commercial versions no longer maintained, very slow or not easily available to use in Python [27].
PEMO-Q	Found implementations cannot be used in Python project structure [28].
HAAQI	High computational complexity makes runtimes unfavorable [29].
STOI and ESTOI	Computationally inexpensive, appears to be affected greatly by higher RTs [30]

TABLE I: An overview of the quality and intelligibility measures considered for use with their reasons for inclusion or exclusion.

audio and a RIR undergoes to estimate the T_{60}. A copy of the repository containing all code used in the project is available on the 4TU repository [31].

A. Acquiring and generating materials

To run an intrusive measure, input requires a clean and degraded audio sample. We create the latter by combining the clean audio with a RIR, as is shown by the noise model in equation 1.

$$x(n) = s(n) * h(n) + v(n) \tag{1}$$

Here, the clean signal $s(n)$ is convolved (denoted by $*$) with the room impulse response $h(n)$ to create the 'degraded' signal $x(n)$. $v(n)$ denotes additive noise, such as Additive White Gaussian Noise (AWGN), that may also be present. When creating the training set, we set the amount of additive noise $v(n) = 0$. During testing, some sets containing noise are created, which is further expanded upon in Section III-D.

To create a training set, we obtain samples of anechoic audio samples from the TSP speech database and simulate a set of RIRs [32]. The TSP dataset is used because it contains a set of speakers of different genders and some non-native speakers. The generated set is divided into a training and testing set with a ratio of 9:1 of train-vs-test. We also create an additional testing set from real RIRs with the same method, but this is further explained in Section III-D.

For this project, we use audio sampled at a rate of 48 kHz. This is done such that the full range of human hearing can be captured properly according to the Nyquist sampling rate [33].

1) Acquiring Room Impulse Responses (RIRs): RIR simulation is done using Habets' method in MATLAB [34]. We choose to simulate RIRs rather than use a dataset of real, measured ones because of the large amount of data necessary to train a MLP: many datasets of real RIRs cover only a few room configurations, meaning that a wide variety of RT values cannot be obtained without combining many sets. Simulation instead provides the ability to easily manipulate the room conditions, allowing us to generate a wider variety of RTs.

Habets' algorithm is based on the Mirror Image Source Method (MISM), a geometric approach to simulating RIRs originally proposed by J. Allen et al. [35]. We generate 181 sets of 100 RIRs each, where sets each have a requested T_{60} value incrementally increased from 0.1 to 2 seconds with steps of 0.01 and a corresponding room configuration. To use Habets' implementation, the following input parameters have to be determined: room dimensions, reflection coefficients of the walls, and microphone and speaker positions. These are determined as follows for each requested T_{60}:

Room dimensions are chosen from 1.5 m and adjusted exponentially using the desired T_{60} and a maximum random deviation of ±0.5 m. The deviation allows for a wider variety of room configurations. The inverse of Sabine's formula, of which the original is shown by Equation 2, is used to obtain an average absorption coefficient. The individual absorption coefficients are then alternated between positive and negative with random deviation, averaging out to the average absorption coefficient. These are then converted to reflection coefficients. The alternation is used as it increases the similarity of the simulated RIRs to real ones.

$$T = \frac{0.161V}{A} \tag{2}$$

Above, T denotes the reverberation time in seconds, V the volume of the setting in m^3, and A the total absorption of the setting in *sabins*, which is equal to the total spatial area of the room S in f^2 multiplied by the average absorption coefficient $\overline{\alpha}$ [2]. It has to be noted that this formula is only an estimate of the RT, and that the RT also depends on speaker and microphone positions.

The speaker and microphone positions are randomly sampled from a uniform distribution with the following restrictions: neither speakers nor microphones may be positioned closer than 0.5 m from a surface, and microphones may not be placed closer than 0.2 m to a speaker. These restrictions are necessary for obtaining consistent results, as stated by the standard for measurement of room acoustic parameters [4]. Per set, 20 microphones and 5 speakers are randomly placed, resulting in 100 combinations between microphone and speaker and thus, 100 RIRs being simulated per room. These RIRs are first exported as arrays from MatLab and then saved as .wav files using a Python script. Here, they are also sliced such that the peak of the RIR, representing the direct signal, is at time $t = 0$.

2) Ground Truth: Because the method generates the input parameters stochastically and uses an estimation formula, the actual T_{60} of the resulting RIRs may differ from the requested one. This is why we run the sets through a script that used Pyroomacoustics' 'measure_rt60' function to measure the T_{60} value to use as ground truth in our model [36]. This function uses Schroeder's curve to estimate the T_{60} and can specify the decay curve to measure to estimate the value from [5]. This method is widely used in practice despite struggling when the decay range available is less than 35 dB [14]. In our project, we call the function to measure the T_{60} using default parameter values, extrapolating from 30 dB instead if the impulse response is too short to do the former.

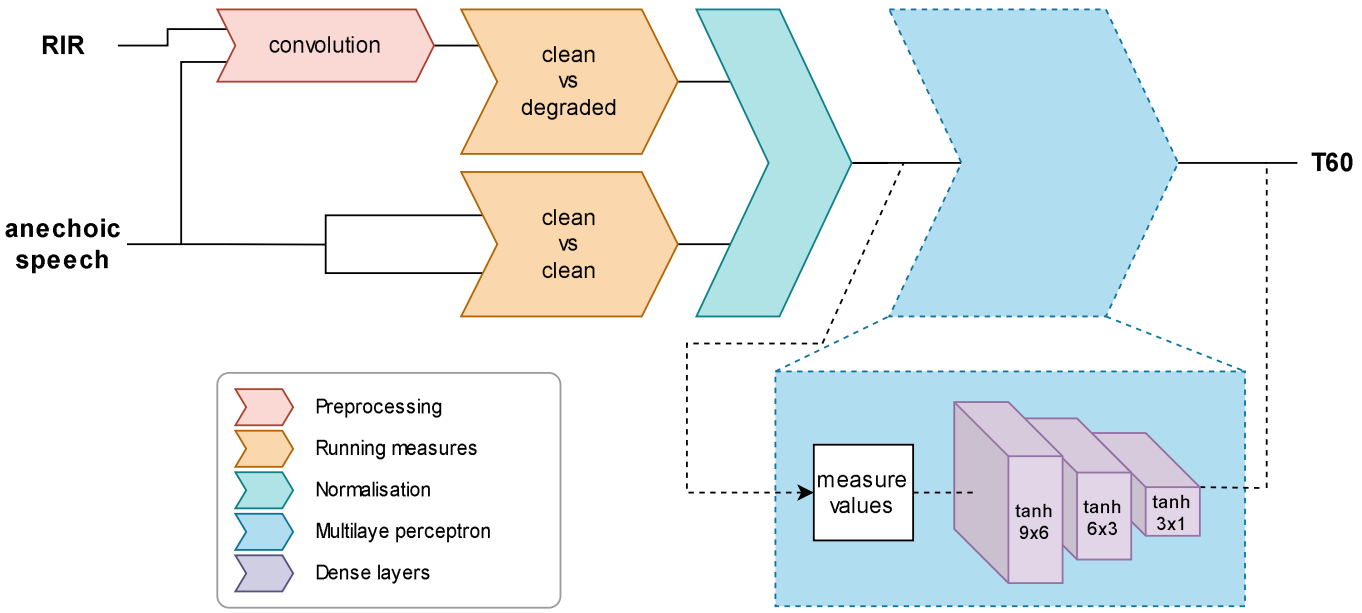


Fig. 1: The layout for the entire pipeline. Clean, anechoic speech is convolved with a room impulse response to create degraded audio. This is then run through the measures together with the clean data and the clean data is also run against itself. The output of the latter is used to normalise the former before the resulting normalised measure values are fed into the multilayer perceptron. The output is an estimate of the T_{60}.

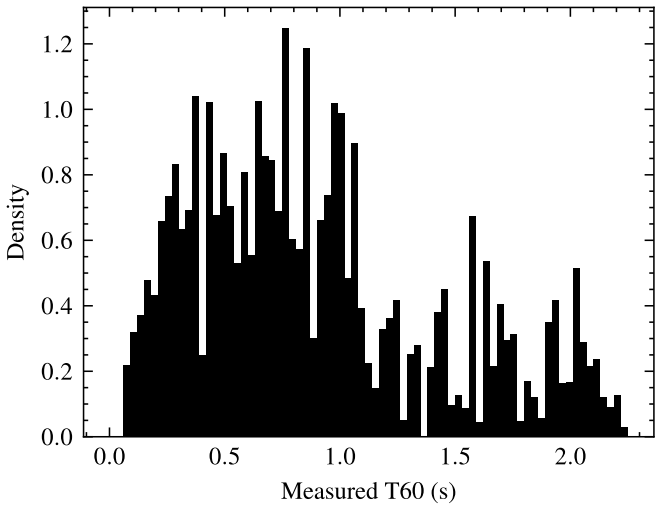


Fig. 2: Distribution of the RIRs generated for the training set. Although requested T_{60}s were uniformly distributed, the T_{60}s measured from the generated sample deviated a lot from the requested.

For simplicity, we assume the measured and estimated T_{60}'s to be the average across all frequency bands or frequency-independent. Table 2 shows the distribution of the T_{60} values of the RIRs generated.

3) Generating degraded audio: We then create the degraded audio segments by convolving the RIRs with the clean audio segments using SciPy's method for FFT convolution [37]. The sampling of clean audio files is done per requested T_{60} bin, taking 7 samples from the clean dataset and 20 RIRs per bin and convolving all of them to create 140 samples per requested T_{60} and a total of 25, 300 degraded audio samples to train and test on. SciPy's 'fftconvolve' function is used for convolution as it has a lower running time than standard time-domain convolution on larger datasets [37]. The end of the resulting degraded audio is trimmed to have the degraded audio length match that of the clean audio, which is necessary to run certain measures. The final audio dataset consists of tuples of the clean and degraded audio samples, labeled with information of the RIR used to create the degraded sample, including the RT T_{60} value.

B. Generating the dataset

To obtain the values of each measure for each tuple created in the previous step, we run the audio dataset through a pipeline that calculates the score for each combination of clean and degraded audio. The pipeline runs through the dataset in batches of 200 samples, running each measure on each sample before saving the intermediate result. By doing this in batches, less memory is required at one time to run the pipeline and intermediate saving is possible. The chunks are concatenated into one final dataset at the end. Aside from obtaining a clean-vs-degraded dataset, each clean audio sample used is also run against itself. We then use the resulting measure values of this set to normalise the clean-vs-degraded set.

C. Creating a regression model

Lastly, we use the feature set generated in the previous section to train a machine learning (ML) model. For this, we use TensorFlow's Keras API to model a MLP regression network [38]. To prepare the features for input, normalisation is applied on each column independently. This allows for a more stable performance of the model once it is trained.

Parameter	Available options
Number of neurons per layer	integer between 1 and 12
Activation function of neurons in hidden layers	['tanh','relu']
Learning rate	[0.01, 0.001, 0.0001]

TABLE II: Overview of the possible parameter values allowed during hypertuning of the multilayer perceptron models. The tuner searches through the option with an incrementally increasing number of epochs to efficiently find the optimal parameters.

1) Model architecture: With the aim for simplicity in model structure in mind, we choose to use a feed-forward ANN which performs regression, also known as a MLP. These networks are known for being able to learn non-linear data well. Another model option is using a CNN but we choose against using these as these models are more suited for higher dimensionality inputs like audio and images to extract patterns while here, we have only five input features.

Using TensorFlow's Keras library [38], the network is built with 5 input features, namely the scores of the measures, and one output representing the estimated T_{60} value. We create 7 different architectures with the number of layers ranging between 1 and 4 which will each be tuned and trained.

2) Parameter tuning and training: Before training the model, hyperparameter training is done to find the optimal network configuration. We use the Hyperband tuner [39] from the 'keras_tuner' package to search for the best activation function, loss function, learning rate, and number of neurons per layer. The possible options to search through are shown in Table II. We configure it to run thrice for each parameter set so that the penalty of unlucky random weight allocation at the start of a search is limited. Additionally, we add early stopping to speed up the process.

After tuning, we train and test the best configuration of each model. The training is done for 100 epochs, allocating a random 10% of the training set as a validation set and shuffling the training set before each epoch. We use the Adam optimiser with the learning rate found from tuning [40]. The trained models are saved to '.keras' files.

D. Testing the models

Once trained, the models are tested on four sets of audio: a set with simulated RIRs, the same set with AWGN added with signal-to-noise ratio (SNR)s equal to 10, 20, 30 and 50 dB, and a test set made using measured RIRs from the dEchorate set [41]. The size of the sets using simulated RIRs is 5096 and that of the real RIR set is 193. Testing on a set of real, measured RIRs is done to determine how representative the training set is for real data and how generalisable and applicable our model is to unseen data. Testing on noisy variants of the simulated test set is done to explore how robust the network is to noise when the RIRs are more similar to those in the training set.

When testing, we measure the MSE, Mean Absolute Error (MAE), and Pearson's correlation coefficient ρ. These metrics are defined according to the equations 3, 4 and 5.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|^2 \tag{3}$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4}$$

When using equations 3 and 4, n refers to the number of samples in the test set, where y_i is the true T_{60} value and $\hat{y}_i$ is the value predicted by our model.

$$\rho(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(X_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2}\sqrt{\sum_{i=1}^{n}\left(Y_i - \overline{Y}\right)^2}} \tag{5}$$

When using Equation 5, X will refer to the ground-truth T_{60} values and Y to the values predicted by the model. The coefficient obtained lies between -1 and 1, where $\rho = \pm 1$ indicates a total positive or negative correlation and $\rho = 0$ indicates no correlation at all.

Additionally, we perform SHapley Additive exPlanations (SHAP) analysis on the best-performing model with real RIR set to explore the contribution of each input using the SHAP package's 'shap.PermutationExplainer' class [42]. The best-performing model is determined by taking the weighted average of the MAE where the real RIR test set weighed twice as much as the simulated ones.

IV. Responsible Research

A. Adhering to FAIR principles

We keep the FAIR and Open Science principles in mind when progressing through this research [43]. As such, the code base is publicly available on the 4TU repository under a GPL license and certain steps have been taken to enforce these principles.

The code base is created with documentation for almost all functions and a README file for every major part or module, allowing for easier understanding of the code written. These are all written aimed at those not yet very familiar with the subject matter. Additionally, the parameters with which we run scripts are provided in separate files or as default values for function parameters, to allow for faster running of the project and improve reproducibility.

To improve interoperability the code base is designed to be as modular and self-contained as possible, such that different modules may run by themselves and can be edited and adapted easily. This is done with the exception of an overarching constants file that keeps track of values and locations used throughout the project.

To further re-usability, the licenses under which the measures used are available are taken into account when choosing the final subset of measures to be used. For example, we choose against using the newer ITU-R recommendation replacing PESQ, POLQA [15], as this is not available under an open license.

B. Bias in datasets

When using machine learning, bias has to be considered when selecting training and test data; implicit bias may always be present when manually selecting data or datasets. In this project, our subjective understanding of quality may influence our choice in clean speech audio and the dataset to use. To mitigate the presence of this bias within our datasets, multiple datasets were considered that had a larger number of speakers with a variety of backgrounds. Aside from the chosen anechoic audio dataset, TSP speech, we also look at the EARS and TIMIT dataset [44], [45]. Additionally, random sampling from the entire clean audio set is done to decide on the subset to use as clean audio, such that our implicit bias on speakers is eliminated.

V. Results

Following the methodology outlined in the previous section and using the minimisation of the MAE as our primary metric, this section entails the results obtained.

We choose to omit the results for the noisy test set with SNR = 50 dB in analysis and the weighted average as the models all perform almost exactly the same here as for the noiseless simulated test set. The scores obtained for this test set are still visible for the sake of completion.

A. Finding the optimal architecture

Table III shows the weighted average defined in Section III-D for the different performance metrics used. From this, we identify that the models 'mae_3div' and 'mae_2div' perform best with both their average MAE being 0.379. We take the first model as the optimal architecture because of its performance in prediction on the real RIR test set, dEchorate, and the average Pearson's correlation coefficient ρ being slightly higher than other models, indicating that this model may encapture the relationship between the input measures and T_{60} slightly better than the other models.

Additionally, models with a decreasing number of neurons over the layers score better than their counterparts that have a uniform number of neurons over the hidden layers. For example, model 'mae_2div' scores 0.493 and 0.972 lower in the MAE and MSE than those of model 'mae_2u'.

Model	Architecture	Activation function	MAE	MSE	ρ
mae_1	5x17x1	tanh	0.475	0.419	0.626
mae_2div	**5x8x4x1**	**tanh**	0.379	0.260	0.551
mae_2u	5x18x18x1	relu	0.872	1.232	0.310
mae_3div	**5x9x6x3x1**	**tanh**	0.379	0.274	0.574
mae_3u	5x17x17x17x1	relu	0.788	1.003	0.354
mae_4div	5x17x13x9x5x1	relu	0.770	1.005	0.311
mae_4u	5x15x15x15x15x1	relu	0.660	0.787	0.541

TABLE III: This overview contains the weighted average scores of the MAE, MSE, and ρ of each model's predictions on the test set, rounded to three decimal places. The average takes equal weighting for the simulated and noisy test sets but gives a weighting twice as high for the real RIR test set. The best-performing models have been highlighted.

B. Performance of model 'mae_3div'

Table IV displays the MAE, MSE and ρ scores calculated from predictions by the model 'mae_3div' when tested on the different test sets. The model is able to attain an MSE of 0.049 on the simulated RIR test set and 0.065 on the real RIR data set, dEchorate. These values are comparable with some of the models' performance in the ACE challenge, though those scores include tests with noise conditions as well [8]. An interesting observation is that, although the dEchorate test set has a higher MAE and MSE than the simulated RIR test set, the maximum negative error for this set is much lower with about -0.5 compared to almost -1.5. This may relate to that the dEchorate set's values have a much smaller range between 0.8 and 1 second. It is notable that the value for Pearson's correlation coefficient for the dEchorate set is very low, 0.159, compared to the other test sets. This suggests that the model does not properly capture the relationship between the values of the input measures and T_{60} when real RIRs are involved.

Test set	MAE	MSE	ρ
Simulated	0.144	0.049	0.918
Sim SNR 50 dB	0.144	0.048	0.919
Sim SNR 30 dB	0.292	0.116	0.884
Sim SNR 20 dB	0.551	0.373	0.794
Sim SNR 10 dB	0.894	0.974	0.526
dEchorate	0.197	0.065	0.159

TABLE IV: Resulting MAE (s), MSE (s^2) and ρ scores obtained from testing model 'mae_3div' on the different test sets, rounded to three decimal places.

Testing on conditions where AWGN was added shows that the model is very sensitive to noise. Figure 3 shows the distributions of the errors between the model's prediction and actual T_{60} values. The bars show the maximum positive and negative errors made by the model as well as the medians for each test set. From this, it can be seen that additive noise makes the model overestimate the RT, causing a median overestimation of 0.25 s when the SNR is equal to 30 dB, up to an overestimation 0.97 s when the SNR decreases to 10.

Looking further, into the error at different T_{60} values, Figure 4 shows that the model makes more and bigger errors with higher T_{60} values. We attribute this behaviour to the RIR distribution in the training set, which contains many more RIRs in the range $[0.1, 1]$ than $[1, 2]$, as shown by Figure 2. This behaviour is consistent throughout the predictions of the simulated test sets.

C. Investigating feature importance with SHAP

In an effort to gain a better understanding of how the model may work internally, SHAP analysis is performed. Figure 5a and 5b show summary plots of the SHAP values calculated. These show the contribution of each input feature on the predicted T_{60} value on the x-axis, where colour indicates the value of the input measure.

Being ordered by global importance within the test set, the STOI measure is shown to have the greatest impact on the predictions made in both sets. In the simulated test set,

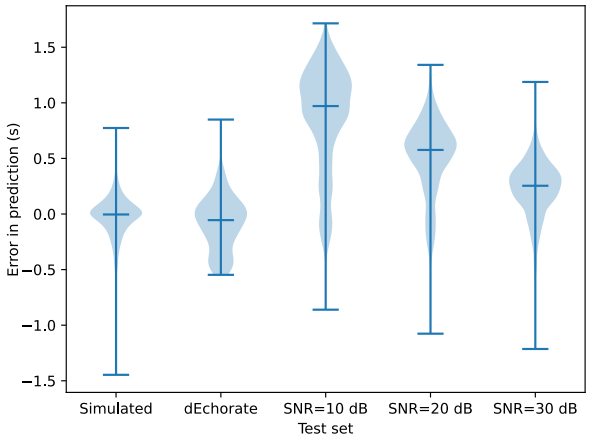


Fig. 3: Violinplot showing the distribution of the prediction errors made by the MLP split per set. The bars show the extremities of the distribution and the median value, and the area indicates the distribution of the errors.

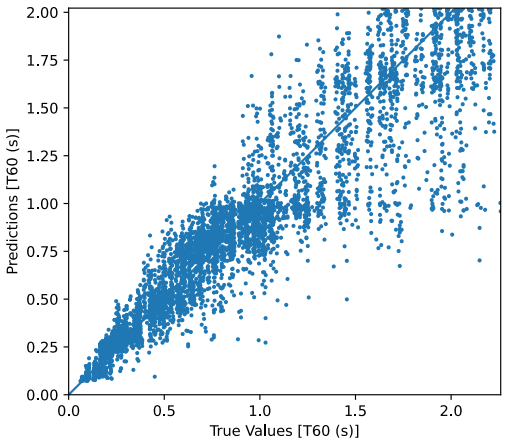


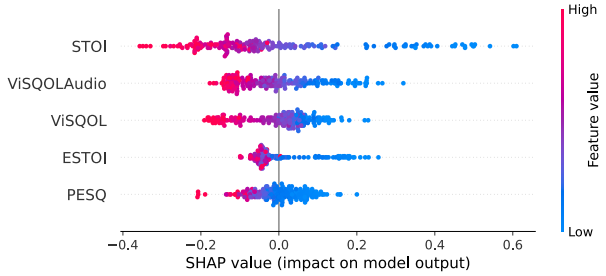
Fig. 4: Comparison between the predicted T_{60} and the ground truth value for the simulated test set. The diagonal line along $y = x$ indicates where the error between prediction and the actual value is equal to 0.

relatively low values of the STOI measure have a very positive impact on the predicted T_{60}, while higher values of STOI have a negative impact on the predicted value, though not as severe as the low values. In the real RIR test set, low and high STOI values have equal impact.

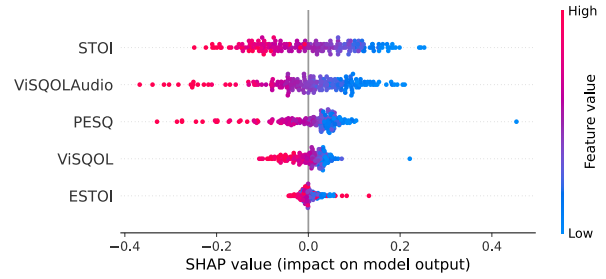
The second most influential input feature in both sets is the ViSQOLAudio measure. Although the contribution of lower ViSQOLAudio values are similar in both sets, the higher values of the measure seem to have a higher contribution in the real RIR test set's predictions.

The other three measures' contribution differs between the simulated and real RIR test set.

In the simulated predictions, ViSQOL has a visibly higher contribution to the output value, showing a negative



(a) Simulated



(b) Simulated

Fig. 5: Summary plot of the SHAP analysis of the simulated and real RIR test sets. The x-axis describes the impact the input value has on the prediction made for a sample, and the colour of the dot indicates the value of the input feature. For a measure, a distribution with high deviations from 0 indicates a high contribution to the predictions while a distribution with low variance centered around 0 indicates a low contribution to model predictions.

relationship with the predicted value. Relatively high values of ESTOI have some but not much impact while lower values have more and PESQ has a fairly balanced contribution between lower and higher values.

Contrastingly, relatively high values of the PESQ measure have a higher impact on the predicted value in the real RIR test set. ViSQOL STOI have a dense distribution at 0, implying that they may not be good measures to use to predict the T_{60} for data with real, measured RIRs.

It must be noted that the contributions identified during SHAP analysis are data-oriented and only describe the contributions in the test sets analysed. Other test sets may have different contributions.

VI. Conclusions and Future Work

In this research, we created a multilayer perceptron (MLP) regression model that tries to estimate the reverberation time (RT), though it falls short of competing with other state-of-the-art estimators, having a mean absolute error of 0.197, a mean squared error of 0.065 and a Pearson's correlation coefficient of 0.159 when tested on a dataset generated from real room impulse responses (RIR). However, from the correlation scores obtained from running the simulated test sets and the SHAP analysis done in Section III-D, we do believe that creating a better estimator using quality and intelligibility measures is possible.

We mainly attribute the lower performance to the quality of the data used to train the network. All models created have only been trained on a dataset generated from simulated RIRs, of which the distribution of the T_{60} values was not uniform, as shown by the distribution histogram in Figure 2. One possible solution to make the training data more representative of real RIRs is to use not only simulated but also real RIRs in training, similar to what Gamper and Tashev have done [7]. As mentioned in section III-A1, obtaining a sufficient number of real RIRs to train a model on is difficult, and an adequate variation in RIRs will be difficult to provide. However, data augmentation methods such as the ones mentioned by Bryan could serve to decrease the size of datasets needed and make them more balanced [46]. Alternatively, further testing on improving the accuracy of the simulated RIRs could be done.

Another way training data quality can be improved may be by increasing the variation in the clean audio datasets could serve to increase robustness. Although the TSP speech dataset is decently varied, datasets created for variation like EARS or non-speech datasets may contribute more to generalisability to multiple audio contexts [32], [44]. Sadly it was out of scope for this project to experiment on the input more.

Lastly, one of the main limitations of our model compared to other state-of-the-art (SOTA) estimators is that it requires clean, non-degraded audio in addition to the degraded one. This makes it much less applicable in many situations. Future exploration on using non-intrusive measures instead of intrusive ones or combining this model with other methods outside of machine learning may be valuable.

References

[1] M. Long, "21 - design of studios and listening rooms," in *Architectural Acoustics*, Elsevier, 2014, pp. 829–871, ISBN: 978-0-12-398258-2. DOI: 10.1016/B978-0-12-398258-2.00021-0.

[2] D. R. Raichel, "Chapter 11. acoustics of enclosed spaces: Architectural acoustics," in *The science and applications of acoustics*, 2. ed, New York, NY: Springer, 2006, pp. 243–279, ISBN: 978-0-387-26062-4.

[3] F. E. Toole, "Chapter 10.1 reverberation," in *Sound reproduction: the acoustics and psychoacoustics of loudspeakers and rooms*, Third edition, New York ; London: Routledge, 2017, ISBN: 978-1-138-92137-5.

[4] *Acoustics - measurement of room acoustic parameters*, 2008. [Online]. Available: https://www.nen.nl/en/nen-en-iso-3382-2-2008-en-123846.

[5] M. R. Schroeder, "New method of measuring reverberation time," *The Journal of the Acoustical Society of America*, vol. 37, no. 3, pp. 409–412, Mar. 1, 1965, ISSN: 0001-4966, 1520-8524. DOI: 10.1121/1.1909343.

[6] H. Kuttruff, "8.4 measurement of reverberation," in *Room acoustics*, 5th ed, London & New York: Spon Press/Taylor & Francis, 2009, pp. 268–272, ISBN: 978-0-415-48021-5.

[7] H. Gamper and I. J. Tashev, "Blind reverberation time estimation using a convolutional neural network," in *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, Tokyo: IEEE, Sep. 2018, pp. 136–140, ISBN: 978-1-5386-8151-0. DOI: 10.1109/IWAENC.2018.8521241.

[8] J. Eaton, N. D. Gaubitch, A. H. Moore, and P. A. Naylor, "Estimation of room acoustic parameters: The ACE challenge," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 10, pp. 1681–1693, Oct. 2016, ISSN: 2329-9290, 2329-9304. DOI: 10.1109/TASLP.2016.2577502.

[9] Y. Liang, S. Li, C. Yan, M. Li, and C. Jiang, "Explaining the black-box model: A survey of local interpretation methods for deep neural networks," *Neurocomputing*, vol. 419, pp. 168–182, Jan. 2021, ISSN: 09252312. DOI: 10.1016/j.neucom.2020.08.011.

[10] *Method for the subjective assessment of intermediate quality level of audio systems*, 2015. [Online]. Available: https://www.itu.int/rec/R-REC-BS.1534-3-201510-I.

[11] M. Torcoli, T. Kastner, and J. Herre, "Objective measures of perceptual audio quality reviewed: An evaluation of their application domain dependence," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 1530–1541, 2021, ISSN: 2329-9290, 2329-9304. DOI: 10.1109/TASLP.2021.3069302.

[12] F. B. Gelderblom, T. V. Tronstad, T. Svendsen, and T. A. Myrvoll, "On the predictive power of objective intelligibility metrics for the subjective performance of deep complex convolutional recurrent speech enhancement networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 32, pp. 215–226, 2024, ISSN: 2329-9290, 2329-9304. DOI: 10.1109/TASLP.2023.3329378.

[13] J. Xia, B. Xu, S. Pentony, J. Xu, and J. Swaminathan, "Effects of reverberation and noise on speech intelligibility in normal-hearing and aided hearing-impaired listeners," *The Journal of the Acoustical Society of America*, vol. 143, no. 3, pp. 1523–1533, Mar. 1, 2018, ISSN: 0001-4966, 1520-8524. DOI: 10.1121/1.5026788.

[14] R. Ratnam, D. L. Jones, B. C. Wheeler, W. D. O'Brien, C. R. Lansing, and A. S. Feng, "Blind estimation of reverberation time," *The Journal of the Acoustical Society of America*, vol. 114, no. 5, pp. 2877–2892, Nov. 1, 2003, ISSN: 0001-4966, 1520-8524. DOI: 10.1121/1.1616578.

[15] I. T. Union, *Perceptual objective listening quality prediction*, 2018. [Online]. Available: https://www.itu.int/rec/t-rec-p.863.

[16] A. Hines, J. Skoglund, A. Kokaram, and N. Harte, "ViSQOL: The virtual speech quality objective listener," in *International Workshop on Acoustic Signal Enhancement*, Germany, 2012, pp. 1–4, ISBN: 978-3-8007-3451-1.

[17] A. Hines, E. Gillen, D. Kelly, J. Skoglund, A. Kokaram, and N. Harte, "ViSQOLAudio: An objective

audio quality metric for low bitrate codecs," *The Journal of the Acoustical Society of America*, vol. 137, no. 6, EL449–EL455, Jun. 1, 2015, ISSN: 0001-4966, 1520-8524. DOI: 10.1121/1.4921674.

[18] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "A short-time objective intelligibility measure for time-frequency weighted noisy speech," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, Dallas, TX, USA: IEEE, 2010, pp. 4214–4217, ISBN: 978-1-4244-4295-9. DOI: 10.1109/ICASSP.2010.5495701.

[19] J. Jensen and C. H. Taal, "An algorithm for predicting the intelligibility of speech masked by modulated noise maskers," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 11, pp. 2009–2022, Nov. 2016, ISSN: 2329-9290, 2329-9304. DOI: 10.1109/TASLP.2016.2585878.

[20] *Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs*, Feb. 2001. [Online]. Available: https://www.itu.int/rec/T-REC-P.862-200102-W/en.

[21] K. Kokkinakis and P. C. Loizou, "Evaluation of objective measures for quality assessment of reverberant speech," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic: IEEE, May 2011, pp. 2420–2423, ISBN: 978-1-4577-0538-0. DOI: 10.1109/ICASSP.2011.5946972.

[22] M. Wang, C. Boeddeker, R. G. Dantas, and A. Seelan, *PESQ (perceptual evaluation of speech quality) wrapper for python users*, version v0.0.4, May 2022. DOI: 10.5281/zenodo.6549559. [Online]. Available: https://doi.org/10.5281/zenodo.6549559.

[23] The MathWorks Inc., *Audio toolbox*, version R2024a, 2024. [Online]. Available: https://www.mathworks.com.

[24] *Python implementation of STOI*, version v0.4.1, 2023. [Online]. Available: https://github.com/mpariente/pystoi/.

[25] T. Fukumori, M. Nakayama, T. Nishiura, and Y. Yamashita, "Estimation of speech recognition performance in noisy and reverberant environments using PESQ score and acoustic parameters," in *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, Kaohsiung, Taiwan: IEEE, Oct. 2013, pp. 1–4, ISBN: 978-986-90006-0-4. DOI: 10.1109/APSIPA.2013.6694136.

[26] T. Kastner and J. Herre, "An efficient model for estimating subjective quality of separated audio source signals," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NY, USA: IEEE, Oct. 2019, pp. 95–99, ISBN: 978-1-72811-123-0. DOI: 10.1109/WASPAA.2019.8937179.

[27] K. Kondo, "On the use of objective quality measures to estimate watermarked audio quality," in *2012 Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, Piraeus-Athens, Greece: IEEE, Jul. 2012, pp. 126–129, ISBN: 978-1-4673-1741-2. DOI: 10.1109/IIH-MSP.2012.36.

[28] R. Huber and B. Kollmeier, "PEMO-q—a new method for objective audio quality assessment using a model of auditory perception," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 6, pp. 1902–1911, Nov. 2006, ISSN: 1558-7916. DOI: 10.1109/TASL.2006.883259.

[29] D. A. M. G. Wisnu, S. Rini, R. E. Zezario, H.-M. Wang, and Y. Tsao, *HAAQI-net: A non-intrusive neural music audio quality assessment model for hearing aids*, Version Number: 4, 2024. DOI: 10.48550/ARXIV.2401.01145.

[30] S. Van Kuyk, W. B. Kleijn, and R. C. Hendriks, "An evaluation of intrusive instrumental intelligibility metrics," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 11, pp. 2153–2166, Nov. 2018, ISSN: 2329-9290, 2329-9304. DOI: 10.1109/TASLP.2018.2856374.

[31] a. Mol, *Codebase underlying the BSc thesis: Using quality and intelligibility measures to create an estimator for reverberation time in an acoustic setting with a multilayer perceptron model*, Jul. 2024. DOI: 10.4121/ddea8296-75d9-4d86-8ba1-fdeb5e4d585d.

[32] P. Kabal, *TSP speech database*, version 2. [Online]. Available: https://mmsp.ece.mcgill.ca/Documents/Data/ (visited on 06/11/2024).

[33] *IEEE standard for advanced audio coding*, ISBN: 9781504498012. DOI: 10.1109/IEEESTD.2023.10258051.

[34] Emanuël Habets, *Ehabets/RIR-generator: RIR generator*, version v2.2.20201022, Oct. 22, 2020. DOI: 10.5281/ZENODO.4117640. [Online]. Available: https://zenodo.org/record/4117640 (visited on 06/10/2024).

[35] J. B. Allen and D. A. Berkley, "Image method for efficiently simulating small-room acoustics," *The Journal of the Acoustical Society of America*, vol. 65, no. 4, pp. 943–950, Apr. 1, 1979, ISSN: 0001-4966, 1520-8524. DOI: 10.1121/1.382599.

[36] R. Scheibler, E. Bezzam, and I. Dokmanic, "Pyroomacoustics: A python package for audio room simulation and array processing algorithms," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, AB: IEEE, Apr. 2018, pp. 351–355, ISBN: 978-1-5386-4658-8. DOI: 10.1109/ICASSP.2018.8461310.

[37] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, "SciPy 1.0: Fundamental algorithms for scientific computing in python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, Mar. 2, 2020, ISSN: 1548-7091, 1548-7105. DOI: 10.1038/s41592-019-0686-2.

[38] TensorFlow Developers, *TensorFlow*, version v2.15.1, Mar. 8, 2024. DOI: 10.5281/ZENODO.4724125. [Online]. Available: https://zenodo.org/doi/10.5281/zenodo.4724125 (visited on 06/03/2024).

[39] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research*, vol. 18, no. 185, pp. 1–52, 2018, Publisher: arXiv Version Number: 4. DOI: 10.48550/ARXIV.1603.06560.

[40] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, Version Number: 9, 2014. DOI: 10.48550/ARXIV.1412.6980.

[41] D. D. Carlo, P. Tandeitnik, C. Foy, N. Bertin, A. Deleforge, and S. Gannot, "dEchorate: A calibrated room impulse response dataset for echo-aware signal processing," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2021, no. 1, p. 39, Dec. 2021, ISSN: 1687-4722. DOI: 10.1186/s13636-021-00229-0.

[42] S. Lundberg and S.-I. Lee, *A unified approach to interpreting model predictions*, Version Number: 2, 2017. DOI: 10.48550/ARXIV.1705.07874.

[43] M. D. Wilkinson, M. Dumontier, I. J. J. Aalbersberg, *et al.*, "The FAIR guiding principles for scientific data management and stewardship," *Scientific Data*, vol. 3, p. 160 018, Mar. 15, 2016, ISSN: 2052-4463. DOI: 10.1038/sdata.2016.18.

[44] J. Richter, Y.-C. Wu, S. Krenn, *et al.*, *EARS: An anechoic fullband speech dataset benchmarked for speech enhancement and dereverberation*, Version Number: 2, 2024. DOI: 10.48550/ARXIV.2406.06185. (visited on 07/01/2024).

[45] Garofolo, John S., Lamel, Lori F., Fisher, William M., *et al.*, *TIMIT acoustic-phonetic continuous speech corpus*, Artwork Size: 715776 KB Pages: 715776 KB, 1993. DOI: 10.35111/17GK-BN40. (visited on 07/01/2024).

[46] N. J. Bryan, "Impulse response data augmentation and deep neural networks for blind room acoustic parameter estimation," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain: IEEE, May 2020, pp. 1–5, ISBN: 978-1-5090-6631-5. DOI: 10.1109/ICASSP40776.2020.9052970.

Appendix

This appendix is a collection of all testing results obtained from all models trained, grouped by model, excluding the best performing model of which the results are shown in Table IV. The subsection names refer to the filename under which the model is saved. Additionally, Table V lists all the architectures of the models after tuning.

TABLE V: Structures of models

Model	**No. neurons per layer (inputx...xoutput)**	**Activation function**
mae_1_3.keras	5x17x1	tanh
mae_2div_3.keras	5x8x4x1	tanh
mae_2u_3.keras	5x18x18x1	relu
mae_3div_3.keras	5x9x6x3x1	tanh
mae_3u_3.keras	5x17x17x17x1	relu
mae_4div_3.keras	5x17x13x9x5x1	relu
mae_4u_3.keras	5x15x15x15x15x1	relu

TABLE VI: Results obtained from testing model 'mae_1'

Test set	**MAE**	**MSE**	ρ
simulated	0.1427	0.0468	0.916
sim SNR 50 dB	0.1430	0.0459	0.917
sim SNR 30 dB	0.4270	0.2560	0.893
sim SNR 20 dB	1.0263	1.2277	0.816
sim SNR 10 dB	1.5816	2.7088	0.696
dEchorate	0.2595	0.0902	0.218

TABLE VII: Results obtained from testing model 'mae_2div'

Test set	**MAE**	**MSE**	ρ
simulated	0.1492	0.0532	0.917
sim SNR 50 dB	0.1480	0.0517	0.918
sim SNR 30 dB	0.2853	0.1156	0.875
sim SNR 20 dB	0.5882	0.4339	0.703
sim SNR 10 dB	0.8093	0.8119	0.491
dEchorate	0.2221	0.0716	0.160

TABLE VIII: Results obtained from testing model 'mae_2u'

Test set	**MAE**	**MSE**	ρ
simulated	0.1415	0.0470	0.922
sim SNR 50 dB	0.1414	0.0461	0.923
sim SNR 30 dB	0.4906	0.3554	0.681
sim SNR 20 dB	1.1600	1.6506	0.234
sim SNR 10 dB	1.8415	3.7998	-0.154
dEchorate	0.7991	0.7699	0.088

TABLE IX: Results obtained from testing model 'mae_3u'

Test set	**MAE**	**MSE**	ρ
simulated	0.1394	0.0455	0.923
sim SNR 50 dB	0.1397	0.0447	0.923
sim SNR 30 dB	0.4669	0.3225	0.742
sim SNR 20 dB	1.0405	1.3361	0.343
sim SNR 10 dB	1.5981	2.9522	-0.238
dEchorate	0.7419	0.6797	0.178

TABLE X: Results obtained from testing model 'mae_4div'

Test set	**MAE**	**MSE**	ρ
simulated	0.1425	0.0472	0.921
sim SNR 50 dB	0.1422	0.0461	0.922
sim SNR 30 dB	0.4725	0.3206	0.738
sim SNR 20 dB	1.0729	1.4513	0.243
sim SNR 10 dB	1.6347	3.1593	-0.353
dEchorate	0.6482	0.5267	0.157

TABLE XI: Results obtained from testing model 'mae_4u'

Test set	**MAE**	**MSE**	ρ
simulated	0.1427	0.0468	0.921
sim SNR 50 dB	0.1430	0.0459	0.922
sim SNR 30 dB	0.4270	0.2560	0.807
sim SNR 20 dB	1.0263	1.2277	0.632
sim SNR 10 dB	1.5816	2.7088	0.628
dEchorate	0.3909	0.2422	0.129