Nachtrag

Boris A. Reif

Many thanks go to Henning Schwanbeck, Stephan Werner and Christian Kehling for their great support. Thanks to go to Prof. Brandenburg and Ms Rodegast for their understanding.

Table 4.3: Rooms at RWTH Aachen
The rooms listed here are from measurements taken at RWTH Aachen. They are part of of the AIR data set which comprises of 344 RIRs; see: [35]. These measurements were taken in various rooms; four of which are labelled with their respective room dimensions and are being used here.

Room	**Volume in m^3**	**Length in m**	**Width in m**	**Height in m**
Studio Booth	26.46	1.8	4.9	3.0
Office	92.8	6.4	5.0	2.9
Meeting Room	124.0	8.0	5.0	3.1
Lecture Room	412.0	10.9	10.8	3.5

Table 4.4: Rooms at TU Ilmenau where RIRs were measured.
The rooms listed here are sorted by building and by their volume given in metres cubed. All of the rooms are on the campus of TU Ilmenau. The volumes stated may slightly differ from the volumes measured by [30]. This is due to measurement and rounding errors. Although measurements were done with a high precision laser pointer, all measurements have to be considered as rough estimates as different results are obtained depending of where exactly in the room the measurement is taken. There are also inaccuracies in the buildings themselves which explains slight deviations in the ceiling height or room width.

Room	**Volume in m^3**	**Length in m**	**Width in m**	**Height in m**
He1519	118	7.3	4.9	3.3
He1520a	118	7.4	5.0	3.2
He2506	143	9.6	4.8	3.1
He2509	146	9.6	4.9	3.1
He2507	146	9.6	4.9	3.1
He1520b	156	9.6	5.0	3.3
He1527	158	9.6	5.0	3.3
He1518	162	9.7	5.1	3.3
K2035	145	9.6	4.9	3.1
K2077	145	9.6	4.9	3.1
K2002a	145	9.6	4.9	3.1
K2032	147	9.7	4.9	3.1
K2026	148	9.8	4.9	3.1
K2003b	151	9.7	5.0	3.1
K2039	189	12.5	4.9	3.1
K2002b	228	14.7	5.0	3.1
K2003a	228	14.7	5.0	3.1
HU011	151	7.5	6.7	3.0
HU117	185	8.1	7.6	3.0
HU204	185	8.1	7.6	3.0
HU010	211	11.0	6.4	3.0
HU012	214	9.5	7.5	3.0
HU210	215	9.7	7.4	3.0
HU013	215	9.7	7.4	3.0
HU201	222	10.0	7.4	3.0
HU129	223	10.0	7.4	3.0
EAZ1220	162	8.5	5.6	3.4
EAZ1221	163	8.4	5.7	3.4
EAZ1223	162	8.5	5.6	3.4
EAZ1222	165	8.5	5.7	3.4

As can be seen from fig. 4.4 the data set contains small to medium sized rooms. The data sets are visualised in four figures which show the transmitter and receiver positions and room sizes and give further explanations; see figs. 4.5 to 4.8.
In addition to the four synthetic base data sets and their expansions a data set of real rooms has also been compiled. This can be seen in table 4.4. The rooms in this data set are small to medium sized lecture halls which fall into roughly the same dimensions as the synthetic rooms. Their RIRs have been convolved with speech. This data set was originally produced and compiled as part of ongoing research by the university. In addition four more real rooms have been added from the AIR data set; see table 4.3 and [35]. All data that was fed into the CNN was also weighted with noise. Every audio signal has an SNR uniformly distributed between 30dB and 50dB. Further each data set was roughly split in a 7:2:1 ratio into a training set, validation set and test set. As required when the model was evaluated with the test set the network had not seen the test set before.

Chapter 5

Experiments and their results

In this chapter a quick overview of the various experiments that have been conducted is given along with a detailed presentation of their results. In this first section the CNN was trained on the base data sets with their respective expansions. For each run the MSE distribution is plotted which serves as a comparison to the work done by Yu and Kleijn [79]. In addition the performance is compared between these experiments in terms of the RMSE, bias and variance per room dimension, i.e. length, width and height. These metrics are taken off the test set and allow for a good comparison to the works by others including groups that have used entirely different methods to estimate the room geometry. In the tables listing these metrics a positive sign signals that the prediction is higher than the true value; a negative value means the predictions is below the true value. There is a different value for each room dimension and their ranges differ, since the length, width and height of each room are estimated independently. As can be seen from these plots: for most experiments the MSE distribution is tailed which shows that the majority of estimation errors are relatively small. However, for the experiments conducted at a learning rate 0.001 it can be seen that for some results the bias is slightly lower or higher than the variance. This is reflected in a curve with a shorter tail and indicates that the model can be improved, for example by decreasing the learning rate.

5.1 Experiment with fixed parameters

Table 5.1: Comparison of variants of the ALLFIXED data set
This table shows the RMSE, bias and variance for each experiment. Each metric has three values since we estimate the room dimensions independently. The last two columns are done with *continuous audio*. Due to all room parameters being fixed the variance is very low in this test.

Data Set	**ALLFIXED**	**with speech**	**with noise**
RMSE (m)	0.2776 0.1507 0.3364	0.2827 0.148 0.329	0.283 0.1488 0.3303
Bias (m)	-0.2491 -0.0063 0.2757	-0.2566 -0.0155 0.2679	-0.2565 -0.0139 0.2693
Variance (m^2)	0.015 0.0227 0.0372	0.0141 0.0217 0.0365	0.0143 0.022 0.0366

In this run the ALLFIXED data set was used for training, validating and testing. In this data set the room coefficients are set to 0.5 and the source source and receiver are fixed;

see appendix B. As can be seen in table 5.1 As with all the main tests on the full data sets the learning rate was set to 0.001 which is the default learning rate in Keras. Since all the parameters are fixed in this first experiment the variance is very low as can bee seen from table 5.1. For the ALLFIXED data set without any noise added one would expect a faster drop in the MSE curve as depicted in fig. 5.1. However, at a learning rate of 0.001 the bias is still too high. This model is good but not ideal. From table 5.1 we can see that we can estimate the width of a room with an accuracy of up to 15 centimetres. The height with an accuracy of about 30 centimetres and the length around 27 centimetres. These values are not as good as expected. Since Yu and Kleijn [79] have shown that it is possible to estimate these values with an average accuracy of up to 4 centimetres. It appears that having a very low variance in the data set is not necessarily an advantage. In fact, from [9] we know that more noise can actually improve the training and lead to better results. From this first experiment we can conclude that simply fixing some

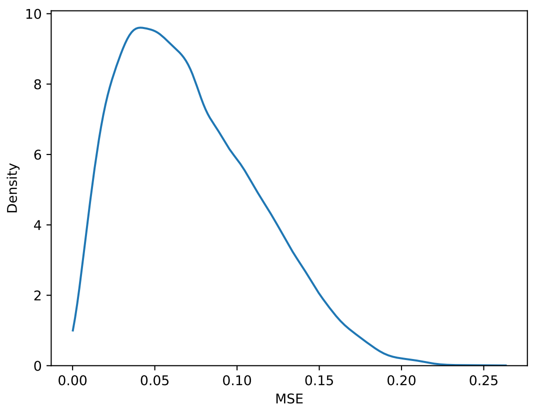

Figure 5.1: ALLFIXED basic data set
This plot shows the error distribution for the ALLFIXED data set. The learning rate α is 0.001. As can be seen in this density plot the majority of rooms is predicted with an error of 0.03 to 0.06 square meters. This result is good but not as expected. The plot shows that the many small errors add up and that in this model we have too many small errors. Compared to [79] the curve drops off too slowly. From table 5.1 we can see that the bias is much larger than the variance.
by *Boris A. Reif*, *License:* 🅭🅯🄏

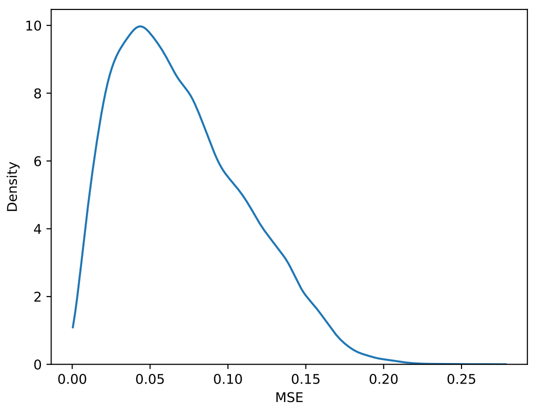

Figure 5.2: ALLFIXED data set convolved with speech
This plot shows the error distribution for the ALLFIXED data set when convolved with speech. In comparison to fig. 5.1 it is noticeable that the error distribution curve is slightly better all be it not much. This run was conducted with continuous audio. From table 5.1 we can see that this model is better. At least for the width the bias is very low; and so is the variance.
by *Boris A. Reif*, *License:* 🅭🅯🄏

of the room parameters such as the position of the source and receiver or the room

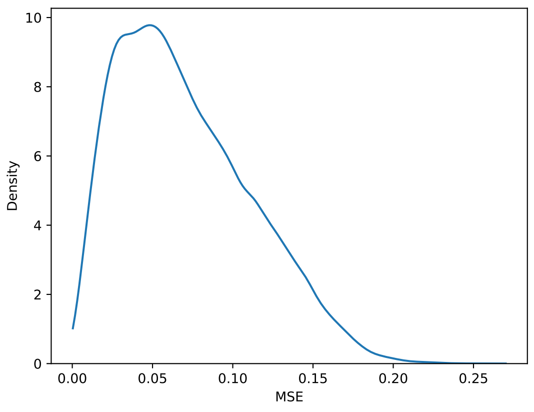

Figure 5.3: ALLFIXED data set convolved with babble noise
This plot shows the error distribution for the ALLFIXED data set when convolved with babble . In comparison to figs. 5.1 and 5.2 it is noticeable that the error distribution curve is the best of all of them. This suggests that a higher variance is in fact beneficial.
by *Boris A. Reif*, *License:* 🅭🅯🄏

coefficients will not necessarily lead to better room geometry estimation. The hyper-parameters play an important role and have an impact on how well our model trains. So has the training set. The curves for continuous audio are actually slightly better than the MSE distribution of pure RIRs.

5.2 Experiment with almost all parameters fixed

Table 5.2: Comparison of variants of the ALLBUTREFL data set
This table shows the three main metrics for runs of the ALLBUTREFL data set. In this set all room parameters but the reflection coefficients are fixed; see appendix B. This implies that the variance of the data set is still very low; similar to what we can see in table 5.1. Due to the same learning rate and other hyper-parameters this model too is slightly under-fitted.

Data Set	ALLBUTREFL	**with speech**	**with noise**
RMSE (m)	0.2515 0.1436 0.3524	0.2713 0.1272 0.3206	0.2804 0.1469 0.3309
Bias (m)	-0.2252 0.0161 0.2986	-0.2553 -0.0144 0.2689	-0.2543 -0.0122 0.2711
Variance (m^2)	0.0125 0.0204 0.0351	0.0084 0.016 0.0305	0.0139 0.0214 0.036

In this section an experiment is introduced in which all parameters but the reflection coefficients are fixed; appendix B. The number of audio files fed into the CNN and the hyper-parameters, most notably the learning rate $\alpha = 0.001$ stay the same. This is reflected in results which show that the model is slightly under-fitted. That is, the variance is low and the bias is slightly too high. Overall we would like to see the bias and the variance more *in tune*; i.e. the bias ought to be lower. Nonetheless the MSE distribution curves in this model as depicted in fig. 5.4, fig. 5.5 and fig. 5.6 are good; meaning they are of an acceptable shape. The majority of estimation errors is small.

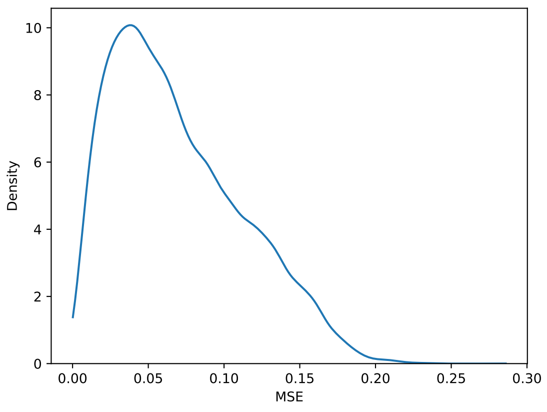

Figure 5.4: ALLBUTREFL base data set
This plot depicts the ALLBUTREFL data set. This contains pur RIRs. The result is similar to [80] albeit not quite as good as the model here is slightly under-fitted; i.e. the bias is slightly too high considering the low variance. This can also be read off in table 5.2.
by *Boris A. Reif*, *License:* 🅭🅯🄏

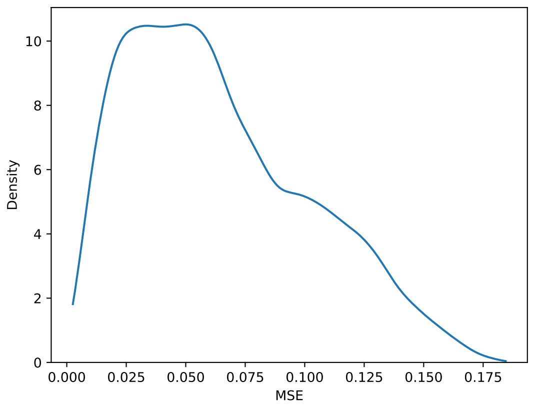

Figure 5.5: ALLBUTREFL base data set with speech
This plot depicts the ALLBUTREFL data set convolved with speech. What stands out is that the curve is basically the same as in the FULLY-BLIND data sets convolved with speech or babble noise. This underlines the importance of regularisation. In comparison this plot shows better results than fig. 5.8. This is as expected, since the ALLBUTREFL data set has a much lower variance.
by *Boris A. Reif*, *License:* 🅭🅯🄏

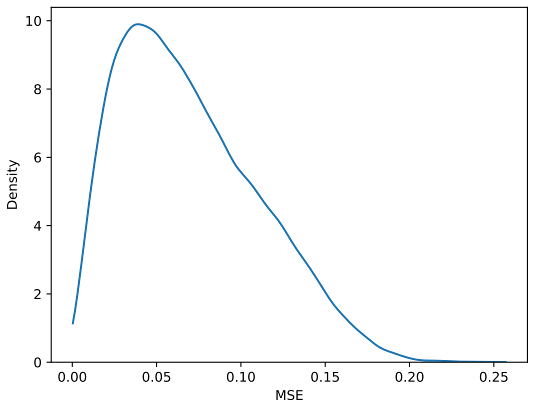

Figure 5.6: ALLBUTREFL base data set with babble noise
This plot depicts the ALLBUTREFL data set convolved with speech. What stands out is that the curve is basically the same as in the FULLY-BLIND data sets convolved with speech or babble noise. This underlines the importance of regularisation. In comparison this plot shows better results than fig. 5.8. This is as expected, since the ALLBUTREFL data set has a much lower variance.
by *Boris A. Reif*, *License:* 🅭🅯🄏

Table 5.3: Comparison of variants of the SEMIBLIND data set
This table shows the main results of the experiment with the SEMIBLIND data set: the variance is very low. The bias is again slighlty too high which means we have an under-fitted model. The learning rate as with the other previous tests was set to 0.001.

Data Set	**SEMIBLIND**	**with speech**	**with noise**
RMSE (m)	0.2607 0.1274 0.331	0.2814 0.1469 0.3273	
Bias (m)	-0.2433 -0.002 0.2806	-0.256 -0.0148 0.2666	
Variance (m^2)	0.0088 0.0162 0.0308	0.0136 0.0214 0.036	

5.3 Experiment with the semi-blind data set

The first two experiments presented so far are mainly important as a verification tool that the methodology presented here is correct. They also serve as a good comparison with [80] and hence as a guide. Although, in these two experiments continuous audio was used the results are of limited practical interest because the assumptions about the room as well as the source and receiver position are not realistic.
In this experiment, with the SEMIBLIND data set the case scenario is more realistic. All room coefficients are chosen at random; see appendix B. The position of the source is random too. The position of the receiver, however, is chosen *semi randomly*. That is, it is around half a meter away from the source and no further away than a meter. This basically simulates a mobile phone scenario: a person is holding a phone with the microphone switched on and the main audio source is the person holding the phone or people around that person talking.

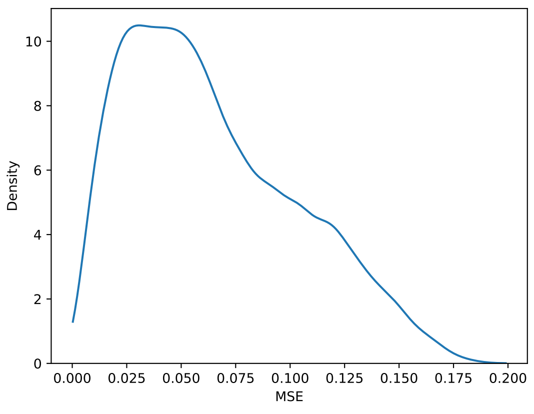

Figure 5.7: SEMIBLIND base data set
This plot depicts the MSE distribution of achieved by training the CNN with the SEMIBLIND data set. Since in the run depicted here pure RIRs were used the result is obviously very good; meaning we have very low estimation errors. The curve has not an ideal shape but a good one; meaning the majority of errors is small. In this case the majority of errors lies around 0.025 o 0.040 metres squared. by *Boris A. Reif*, *License:*

5.4 Experiment with the fully-blind data set

This section is the most important and interesting one so far, as here the most difficult scenario is simulated and examined. All room coefficients are chosen at random as can

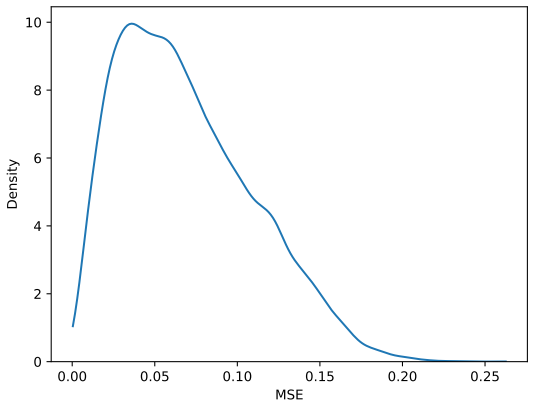

Figure 5.8: SEMIBLIND base data set with speech
This plot depicts the SEMIBLIND data set convolved with speech. Since here we have trained our CNN with continuous audio the result is obviously not as good as in fig. 5.7. However, it is not near as bad as one would maybe expect. From table 5.3 we can see the average error is around 15 to 30 centimetres.
by *Boris A. Reif*, *License:* 🅭🅯🄏

Table 5.4: Comparison of variants of the FULLYBLIND data set
This table shows the results for the main metrics of the experiments with the FULLYBLIND data set. The model which was build with a learning rate of 0.001 and a batch size of 50 like all the previous models we have seen so far is still slightly under-fitted. Its bias is too high. Nonetheless the figures clearly show that it is possible to estimate the room geometry from continuous audio with an accuracy of up to around 20 centimetres.

Data Set	**FULLYBLIND**	**with speech**	**with noise**
RMSE (m)	0.2718 0.1312 0.3229	0.2703 0.1274 0.3201	0.2715 0.1277 0.3218
Bias (m)	-0.2541 -0.0133 0.2694	-0.2541 -0.0135 0.2684	-0.2556 -0.0136 0.27
Variance (m^2)	0.0094 0.017 0.0317	0.0085 0.0161 0.0304	0.0084 0.0161 0.0306

be seen from the code in appendix B. This is also the most practical test since it assumes no prior knowledge here about the source and receiver position within the room. Nor their position to each other. Since all room parameters are chosen at random too it is fair too say that no prior knowledge is assumed here. This is different from many other RGE methods presented in chapter 3 which often assume prior knowledge of either the distance between the source and receiver (as in the previous section) or the room.

As can be seen from fig. 5.10 and fig. 5.11 is appears to make little difference whether babble noise or pure speech is picked up by the receiver. The CNN seems to be little affected by it. It can also handle the added white Gaussian noise (mentioned in chapter 4 quite well. These results mean that even in a nearly worst case scenario, i.e., a noisy audio of people talking in a room, it is still possible to measure the room geometry of that room with an accuracy of around twenty centimetres; see table 5.4. This is with a slightly under-fitted model. Improvements to the model should boost that value by a couple of centimetres more.

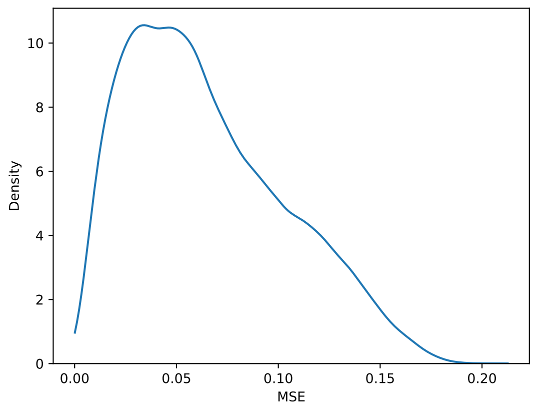

Figure 5.9: FULLYBLIND base data set
This plot depicts the error distribution for the FULLYBLIND base data set. For this plot no audio was used. The majority of erros lies around 0.05 metres squared.
by *Boris A. Reif*, *License:* 🅭🅯🄏

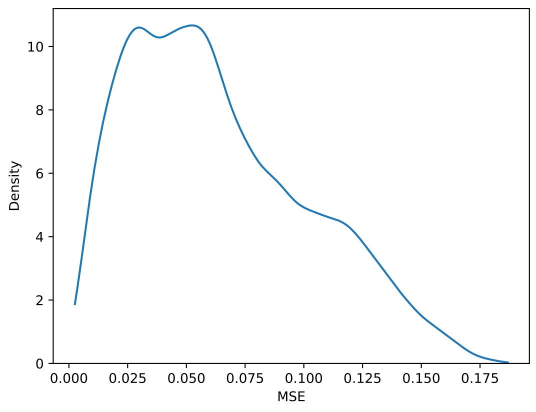

Figure 5.10: FULLYBLIND base data set with speech
This plot depicts the error distribution for the FULLYBLIND base data set convolved with speech. As can be seen the results are again remarkably good. The shape of the curve is not perfect which suggests that a *k-fold cross-validation* strategy or a *split & shuffle cross-validation* strategy would improve the results in this model. However, when comapared to fig. 5.1 the result is again considerably better. What is interesting however, is the comparison with fig. 5.11: it appears that for the CNN it does not make any difference whether babble noise was used or not. The results are practically the same.
by *Boris A. Reif*, *License:* 🅭🅯🄏

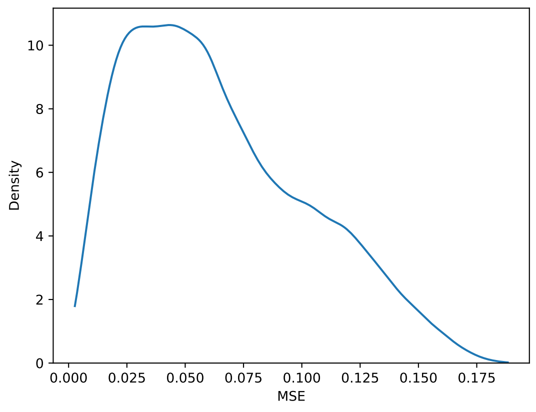

Figure 5.11: FULLYBLIND base data set with *babble noise*
This plot shows the error distribution for the RGE on the FULLYBLIND data set where each RIR is convolved with *babble noise* in order to simulate the cocktail party scenario. Although the babble noise used in this data set is not perfect (only Stephen Fry's voice was used) the results in this data set nonetheless are impressive and exceed expectations. The curve shape is still the same as in fig. 5.1. However, the magnitude of the errors in total is much smaller. This is remarkable since the ALLFIXED base data set consists only of RIRs. This suggests that the type of noise matters a lot when training the data. This result though surprising as it is, is not inconsistent with the literature; see the work by Bishop [9].
by *Boris A. Reif*, *License:* 🅭🅯🄏

5.4.1 Experiment with lower learning rate

Table 5.5: Comparison of variants of the FULLYBLIND data set
This

Data Set	**with speech**	**with noise**
RMSE (m)	0.2716 0.1275 0.3205	0.2718 0.1276 0.319
Bias (m)	-0.2556 -0.0132 0.2688	-0.2557 -0.0142 0.2672
Variance (m^2)	0.0085 0.0161 0.0305	0.0085 0.0161 0.0304

As a result of the previous section another test has been conducted by training the neural network at a lower learning rate. The learning rate is fixed at 0.0008. The results are shown in table 5.5. These results show that the model is still slightly under-fitting. However, the overall results are good. They illustrate that we can estimate the geometry of a room with an accuracy of up to 20 to 30 centimetres. This fares well when compared to [80].

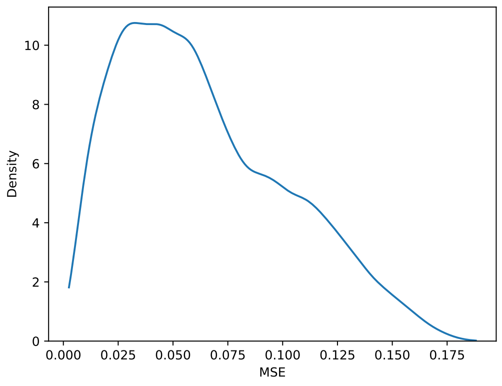

Figure 5.12: FULLYBLIND base data set with speech
This plot shows the MSE distribution for the CNN trained with speech at a lower learning rate.
by *Boris A. Reif*, *License:* 🅭🅯🄏

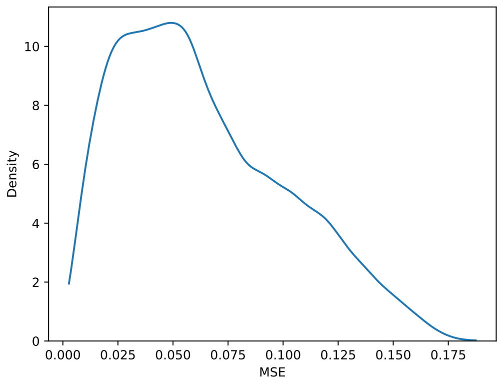

Figure 5.13: FULLYBLIND base data set with babble noise
This plot shows the error distribution of the NN trained at a lower learning with babble noise.
by *Boris A. Reif*, *License:* 🅭🅯🄏

5.5 Experiments with real rooms and continuous audio

Table 5.6: Comparison of the different learning rates
This table shows the different learning rates and their effect on the main metrics used as a comparison tool in this thesis.

α	**0.001**	**0.0008**	**0.00008**
RMSE (m)	1.7257 0.6226 1.4388	1.8725 0.7326 1.5156	1.5869 1.6618 2.1166
Bias (m)	-0.8823 0.0738 1.3299	-1.484 -0.2012 1.1067	-1.2738 -0.0576 1.8051
Variance (m^2)	2.1996 0.3821 0.3016	1.3038 0.4963 1.0721	0.8957 2.7583 1.2214

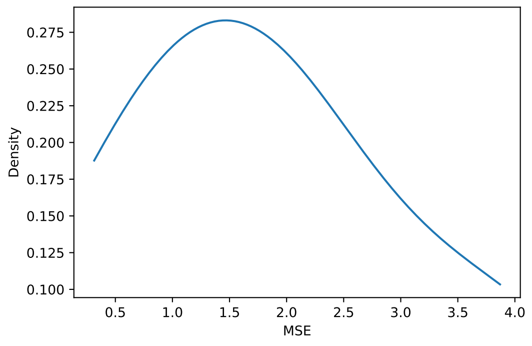

Figure 5.14: Real rooms mixed with synthetic rooms
This plot depicts the MSE distribution curve for training, validating and testing the CNN with an even mix of synthetic rooms and real rooms. In this plot the batch size was 2 and the learning rate was set to 0.001.
by *Boris A. Reif*, *License:* 🅭🅯🄏

So far the discussion has only centred around synthetic rooms generated with RIR generator introduced in section 4.3. In this section real rooms are being used. As already mentioned in chapter 4 good data sets are desirable for machine learning. For estimating the room geometry good data sets that are labelled and easily accessible are still rare and relatively small. In this section the results are introduced by training, validating and testing the CNN with continuous audio recorded in real rooms and synthetic rooms in a 1:1 ratio. As introduced in section 4.3 the data sets by TU Ilmenau and RWTH Aachen are being used since they are freely available. (The TU Ilmenau data set is also relatively large.) Noise is mixed, one person talking, several person talking. Since the overall data set is tiny the batch size size is reduced to two. The even share of synthetic and genuine rooms was chosen so that there is no bias. Obviously due to the small size of the data set one cannot expect much. Nonetheless, fig. 5.14 shows that the curve of the shape is acceptable. The model trained at a learning rate of 0.001 shows that the majority of errors are around 1.5 metres squared. As can be seen from table 5.6 as well as from the figures, the best estimator for this small data set is achieved with a learning rate of 0.0008, depicted in fig. 5.15. However, the data set is much to small to create an estimator of practical use. The results are even worse when training, validating and testing is done on genuine rooms only. This underlines the importance again of producing larger real room data sets that are labelled.

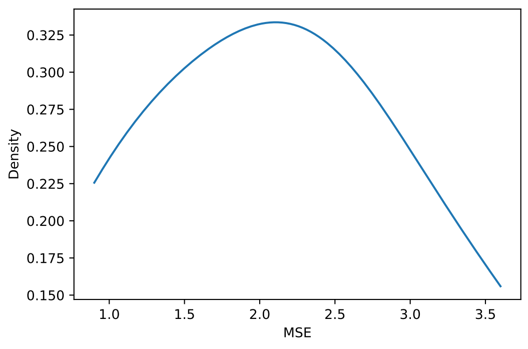

Figure 5.15: Real rooms mixed with synthetic rooms at a learning rate of 0.0008
This plot depicts the MSE distribution curve for a learning rate that was set to 0.0008. As compared to fig. 5.14 this does not really improve our estimator.
by *Boris A. Reif*, *License:* 🅭🅯🄏

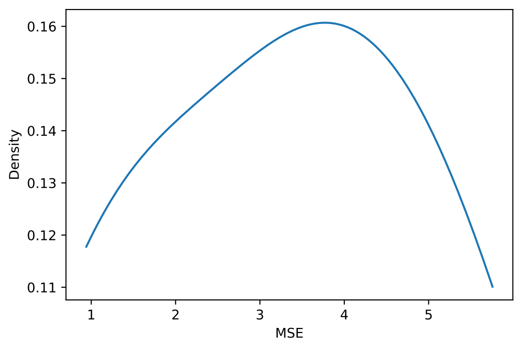

Figure 5.16: Real rooms mixed with synthetic rooms at a learning rate of 0.00008
This plot shows what happens if we further increase the learning rate. In fact in this case our estimator gets even worse. The majority of errors lies now around the four squared metres mark. From table 5.6 we can see that all metrics look bad.
by *Boris A. Reif*, *License:* 🅭🅯🄏

5.6 Experiments with biased data sets and continuous audio

Table 5.7: Comparison of effect of bias in the data set
This table compares the results for biased data sets. The learning rate was kept at 0.001 and the batch size at 2 for all runs.

Bias	**100**	**1000**	**10000**
RMSE (m)	1.2217 0.8171 1.25	1.2137 1.3629 1.7683	1.0775 0.432 0.9072
Bias (m)	-1.0642 0.3016 1.1682	-0.4126 0.54 1.3784	-0.9069 0.024 0.8482
Variance (m^2)	0.3601 0.5767 0.1976	1.3029 1.566 1.2271	0.3385 0.186 0.1035

As we have seen in the previous section labelled data sets of real rooms are very small. This means a meaningful estimation based on these data sets is currently difficult to achieve. This section explores to what extend a small set of genuine RIRs convolved with speech can be used to augment and thus fine tune a synthetically generated data set. fig. 5.17, fig. 5.18 and fig. 5.19 depict the results of this experiment. As can be seen the estimator can be improved by adding more and more synthetically generated audio but obviously this means that the result becomes more and more dependent on the synthetically generated data. Mixing the synthetic and genuinely measured data

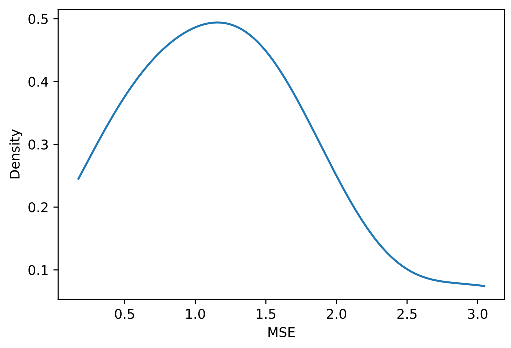

Figure 5.17: 34 Real rooms mixed with 100 synthetic rooms
This plot shows shows the MSE distribution of the CNN depicted in table 4.1 trained, validated and tested on an uneven mix of continuous audio data. The curve is slightly better than what is depicted in section 5.5.
by *Boris A. Reif*, *License:* 🅭🅯🄏

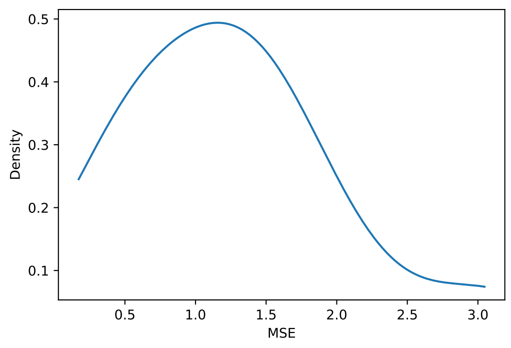

Figure 5.18: 34 Real rooms mixed with 1000 synthetic rooms
This plot shows the curve for an even higher share of synthetically generated continuous audio in the data set. However, when compared to fig. 5.17 we can see that the improvement is minor.
by *Boris A. Reif*, *License:* 🅭🅯🄏

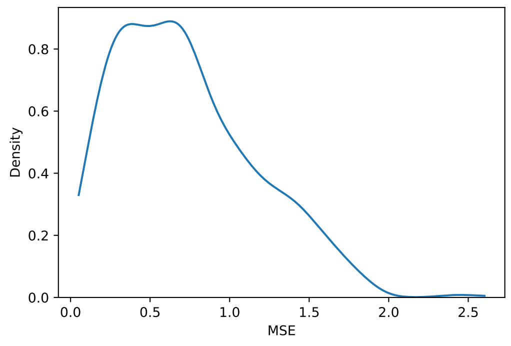

Figure 5.19: 34 Real rooms mixed with 10000 synthetic rooms
This figure shows the MSE distribution for the highest mix of synthetically generated continuous audio. The estimator obviously improves, but is now highly dependent on the quality of the synthetically generated data.
by *Boris A. Reif*, *License:* 🅭🅯🄏

makes thus only sense if the portions between the two do not differ too much. In the data set used here it is easy to see that the difference is simply too large.

The results shown here can be compared to the work by Yu and Kleijn [79]. Yu and Kleijn [79] have already demonstrated that it is perfectly feasible to estimate the room geometry by training a CNN with RIRs. In this thesis the challenge was to train a CNN using continuous audio. As the results above demonstrate (see the figures above) it is not only feasible to estimate the room geometry directly from audio, but with the right model results can come very close to estimating directly from the RIRs. For the CNN the speech signal is merely a signal with a higher noise content. This slightly degrades performance but is not prohibitive in estimating the room geometry from audio signals.

Chapter 6

Conclusions and Future Work

In this thesis it has been shown that the room geometry can be estimated from a continuous audio signal without any prior knowledge of the room or the source and receiver position. Most importantly without prior knowledge of the room's impulse response. With the best model presented here, this estimation can be done with an accuracy of up to twenty centimetres from a noisy recording. It has also been shown that this model can be slightly further improved by reducing the bias.

Immediate future work has to deal with the optimisation of the hyper-parameters. The setting of the hyper-parameters has an enormous effect on the networks ability to learn effectively. Finding the right balance between the variance and the bias proves to be difficult. Another investigation that needs to be done is the effect of noise. It appears that the performance can actually be improved with slightly more noise rather than less. This is consistent with the literature on NN; see Bishop [9].

There are many directions in which the work presented here can be taken. One obvious choice is to train the CNN with spectrograms of the audio signals instead of the audio signals themselves. What results this might bring is subject to controversy. Strictly speaking the spectrograms do not contain any other information. However, there has been much greater progress in CNN architectures for image training than there has been for 1D signals. The challenge however is that for this a new architecture has to be devised and justified. In addition the training periods will extend even longer which makes the work rather cumbersome.

Appendix A

PYTHON scripts for the CNN

Listing A.1: PYTHON code used for scaling the targets (room dimensions)

```
def scale_to(x, x_min, x_max, t_min, t_max):
    """
    Scales x to lie between t_min and t_max
    """
    r = x_max - x_min
    r_t = t_max - t_min
    assert (math.isclose(0, r, abs_tol=np.finfo(float).eps) == False)   ❶
    x_s = r_t * (x - x_min) / r + t_min
    return x_s
```

Listing A.2: PYTHON code used for re-scaling the results back to the original targets (room dimensions)

```
def scale_inv(x_s, x_min, x_max, t_min, t_max):
    """
    Inverse scaling
    """
    r = x_max - x_min
    r_t = t_max - t_min
    assert (math.isclose(0, r_t, abs_tol=np.finfo(float).eps) == False)   ❷
    x = (x_s - t_min) * r / r_t + x_min
    return x
```

Both functions, the scaling function and the inverse scaling function take as first argument the value to be scaled as an input. The last two arguments are the scaling targets that is the new minima and maxima. Note that the minimum and maximum of the input are not computed within the function as would normally be the case. But have to be computed outside the function and are given as arguments. This is necessary because we want all RIRs to be scaled in the same way. Also note ❶ and ❷ in both functions. This is to check that there is indeed an input range. Errors can be discovered by noticing that inputs may just be zero.

Listing A.3: PYTHON code used for building the CNN model)

```python
def get_cnn_model(input_shape):
    """
    Returns the CNN Model
    Takes number of Samples
    """
    #print(input_shape)
    inputs = Input(shape=input_shape, name='Input')
    #print(inputs.shape)
    reshape1 = Reshape((input_shape,-1),name='Reshape_1')(inputs)
    conv1d = Conv1D(filters=32, kernel_size=4, strides=4, name="1st_Conv1D")(reshape1)
    banol1 = BatchNormalization(name='1st_Batch_Normalisation',momentum=0.9)(conv1d)
    leaky1 = LeakyReLU(name='1st_Leaky_ReLU',alpha=0.1)(banol1)
    conv2d = Conv1D(filters=32, kernel_size=2, strides=2, name="2nd_Conv1D")(leaky1)
    banol2 = BatchNormalization(name='2nd_Batch_Normalisation',momentum=0.9)(conv2d)
    leaky2 = LeakyReLU(name='2nd_Leaky_ReLU',alpha=0.1)(banol2)
    conv3d = Conv1D(filters=128, kernel_size=8, strides=8, name="3rd_Conv1D")(leaky2)
    banol3 = BatchNormalization(name='3rd_Batch_Normalisation',momentum=0.9)(conv3d)
    leaky3 = LeakyReLU(name='3rd_Leaky_ReLU',alpha=0.1)(banol3)
    conv4d = Conv1D(filters=128,kernel_size=2, strides=2, name="4th_Conv1D")(leaky3)
    banol4 = BatchNormalization(name='4th_Batch_Normalisation',momentum=0.9)(conv4d)
    leaky4 = LeakyReLU(name='4th_Leaky_ReLU',alpha=0.1)(banol4)
    conv5d = Conv1D(filters=512,kernel_size=2, strides=2, name="5th_Conv1D")(leaky4)
    banol5 = BatchNormalization(name='5th_Batch_Normalisation',momentum=0.9)(conv5d)
    leaky5 = LeakyReLU(name='5th_Leaky_ReLU',alpha=0.1)(banol5)
    conv6d = Conv1D(filters=512,kernel_size=4, strides=4, name="6th_Conv1D")(leaky5)
    banol6 = BatchNormalization(name='6th_Batch_Normalisation',momentum=0.9)(conv6d)
    leaky6 = LeakyReLU(name='6th_Leaky_ReLU',alpha=0.1)(banol6)
    conv7d = Conv1D(filters=1024,kernel_size=4, strides=4, name="7th_Conv1D")(leaky6)
    banol7 = BatchNormalization(name='7th_Batch_Normalisation',momentum=0.9)(conv7d)
    leaky7 = LeakyReLU(name='7th_Leaky_ReLU',alpha=0.1)(banol7)
    conv8d = Conv1D(filters=1024,kernel_size=1, strides=1, name="8th_Conv1D")(leaky7)
    banol8 = BatchNormalization(name='8th_Batch_Normalisation',momentum=0.9)(conv8d)
    leaky8 = LeakyReLU(name='8th_Leaky_ReLU',alpha=0.1)(banol8)
    reshape2 = Reshape((1024,), input_shape=(1,1024), name="Reshape_2")(leaky8)
    dense1= Dense(160, name="1st_Dense")(reshape2)
    dense2= Dense(64, name="2nd_Dense")(dense1)
    length = Dense(1, name='length')(dense2)
    width  = Dense(1, name='width')(dense2)
    height = Dense(1, name='height')(dense2)
    model = Model(inputs=inputs, outputs=[length,width,height], name="RIR_Model")
    return model
```

This is the PYTHON implementation of the CNN architecture described in [80, 79] and listed in 4.1. It is used for all experiments in this thesis. Unlike Yu and Kleijn [80] who used Pytorch for their implementation, KERAS and Tensorflow were used in this thesis. In order to come close to the implementation by Yu and Kleijn [80] the KERAS default values for batch normalisation and leaky relu layers have been changed to the Pytorch values.

Listing A.4: PYTHON code used for generating babble noise

```python
from pathlib import Path
import random
import numpy as np
import wave
def create_babble_noise():
    channels = random.randint(1 + MIN_CHANNELS, 1 + MAX_CHANNELS)
    speech_files = list()
    lengths= list()
    for i in range(0,channels):
        speech_file = random.choice(os.listdir(SPEECH))
        speech = open_wave(SPEECH + speech_file)
        lengths.append(len(speech))
        speech_files.append(speech)

    speech_matrix=np.array([np.pad(v, (0, np.max(lengths) - len(v)), 'constant') for v in speech_files])
    #print(speech_matrix.shape)
    axs=[1]*channels
    roll=list()
    for i in range(0,channels):
        #print(i)
        roll.append(random.randrange(0, speech_matrix.shape[1],SAMPLES))

    # babble_matrix=tf.roll(speech_matrix,shift=roll, axis=axs)
    babble_matrix=np.roll(speech_matrix,shift=roll, axis=axs)
    babble = np.sum(babble_matrix,axis=0)
    babble = np.divide(babble, np.max(babble))
    snd = scale_to(babble,np.min(babble),np.max(babble),-32768.0,32767.0)
    babble_noise = snd.astype(np.int16)
    babble_list=list(divide_chunks(babble_noise, 2**21))
    return babble_list[:-1]
```

This is the main PYTHON code used to generate the babble noise. The number of channels used and the shift are chosen at random. So is the segment that is picked after the noise has been convolved with the RIR. This means that the intensity of the noise might change from wave file to wave file. This simulates a variety of rooms, from more quite to very busy; full of people.