EUROGRAPHICS 2019 / P. Alliez and F. Pellacini
(Guest Editors)

Volume 38 (2019), Number 2

Supplementary Document for Learning to Importance Sample in Primary Sample Space

Quan Zheng and Matthias Zwicker

University of Maryland, College Park, USA
quan.zheng@outlook.com; zwicker@cs.umd.edu

Abstract

Our paper "Learning to Importance Sample in Primary Sample Space" proposes a novel importance sampling approach for Monte Carlo rendering that uses a neural network to learn how to sample from a desired density represented by a set of samples. In this supplementary document, we provide information on four additional experiments:

*1. **Reusing trained neural networks**. Our neural networks trained on one camera view of a scene can generalize to multiple new camera views. Given weights of a trained neural network for one view, we reuse them to initialize a network for the new view. Experimental results show that reusing trained weights can effectively accelerate re-training of up to 90-degree camera view changes. Hence training costs that were spent on previous views can be amortized over new views.*

*2. **Importance sampling small illumination features**. Our approach works in a data-driven manner, and the learning of importance sampling depends on provided training data. Our additional experiments evaluate the performance of neural network importance sampling of small illumination features, with respect to different sizes of training datasets.*

*3. **Efficiency of resampling**. We present an analysis of time costs of the resampling stage, and discuss the size of the candidate set of initial examples in terms of a trade-off between efficiency and quality of resampling.*

*4. **Importance sampling in high dimensional space**. We further investigate how our approach behaves in the case of importance sampling a high dimensional primary sample space.*

1. Reusing Trained Neural Networks

Visible elements of the 3D real world are richly diverse, and the difference in features between two arbitrary scenes can be quite large. Therefore, our approach opts to perform per-view learning, and computational costs have to be invested in the training and inference processes for each camera view. Fortunately, there is coherency of contents between different camera views within a specific scene. Given a new camera view, we can reuse weights of a neural network trained on a previous view as an initialization, instead of training the neural network from scratch. In this way we can accelerate the convergence speed of neural networks training as demonstrated by the following experimental results.[†]

Starting from a camera view 1, we change gaze angles of cameras by approximately 10, 45, and 90 degrees for views 2, 3, and 4, respectively. Figure 3 and Figure 4 compare rendering errors under equal training time with and without network weight reuse from view 1 for two different scenes. This demonstrates that weight reuse significantly decreases error compared to training from scratch, and only a few minutes of training time can reduce the MSE of baseline path tracing by almost a factor of two.

Figure 1 plots test losses of 4D PSS learning in terms of the reusing approach and training from scratch. The size of a training dataset is equal to $160 \times 88 \times epp$. Note that test data are not used in the training process, thus test losses faithfully reflect generalization ability of trained neural networks. The neural network of view 1 is initially trained on an epp-32 dataset until convergence. The results show that weight reuse can lead to much faster convergence compared to training from scratch. For example, we can observe in view 2 that, using less than 100 seconds, weight reuse gives lower errors than training from scratch for 800 seconds. For view 3 and view 4, the coherence with view 1 is greatly attenuated, thus the advantage of weight reuse is gradually reduced. We still observe faster convergence than training from scratch, albeit with higher test errors at the start of training. To sum up, given the same level of test errors, the plots in Figure 1 show an accelerated convergence of weight reuse by factors of roughly 5.0x/4.0x/2.0x for camera views 2/3/4, compared to training from scratch.

Figure 1 further investigates the trade-off between using a smaller training data set and faster training time (per epoch) ver-

[†] All timings in the supplementary document are measured with respect to a 3.60 GHz i7-7700 Intel CPU and a Nvidia Titan V GPU, which is different from the main paper.

sus larger datasets and slower training time (per epoch). The results show that at equal training time, a larger data set (trained over fewer epochs) generally performs slightly better. Finally, we experimented with reusing networks for training on epp-8 datasets, but found that it leads to over-fitting problems.

Figure 3 and Figure 4 show equal sample count (128 *spp*) comparisons of the COUNTRY KITCHEN and the WHITE ROOM scene. We compare images of three new views rendered with baseline path tracing, and the epp-16 weight reuse and epp-16 training from scratch approaches at equal training time. Training sessions stop at the epoch closest to the preset training time of a specific view. The reusing approach is initialized with weights of networks trained on camera view 1. As can be noticed, weight reuse generates images with less noise and lower numerical errors. Compared to view 1, the change of camera gaze angle in view 2 is smaller than that in camera view 3 and 4, thus weight reuse provides more significant improvements in view 2.

Since the coherency among camera views is more significant when they are close, it is advantageous to incrementally reuse network weights optimized on successive views along a camera path, instead of reusing the network weights of a single original view. Figure 2 demonstrates the advantages of this approach. We can see that reusing networks of closer views effectively improves the speed of convergence. This shows that incremental weight reuse is a simple and effective method for rendering multiple frames along a predefined camera path, along which neighboring frames share coherent scene contents.

2. Importance sampling small illumination features

Small illumination features like caustics or highlights, are notoriously challenging for many rendering methods. Figure 5 shows an example scene with such difficulties, including bright cardioid caustics in the silver ring, faint golden caustics (green close-ups), faint white caustic with fine structures (red close-ups) and small glints on coins. We use equal sample budget (128 spp) comparisons between our method and the path tracing baseline. Our method conducts 4D PSS importance sampling. The size of a training dataset is set to *epp* (examples per pixel) times 100×100. Three training datasets are denoted as *epp*-1, *epp*-4 and *epp*-64.

The path tracing baseline applies uniform random PSS sampling, and the probability of sampling paths of caustics is low. Therefore, the baseline is inefficient to render small caustics. In contrast, our neural approaches successfully capture small illumination features. Also, it can be observed that the neural network trained on the *epp*-1 data set gives results close to the baseline path tracing approach. This is because we initialize our neural network with an identity mapping (i.e., a uniform random distribution), and it will fall back to the uniform PSS sampling when training examples are not sufficient. Given more training examples, our neural approaches significantly outperform the baseline. Note that the fine structures of small caustics are clearer and the faint caustics are smoother in our method. In summary, given sufficient training data our approach is able to improve rendering of challenging small illumination features.

Table 1: *Efficiency of resampling. The number of examples in the candidate set is equal to* α *times the number of examples in the training dataset, which is set to* $160 \times 88 \times 16$*. For fair comparisons, timings are with respect to computation of 1 CPU thread.*

Ratio α	Candidates#	Draw candidates (s)	Resample (s)
1	225280	3.86	116.64
2	450560	7.69	118.71
4	901120	15.57	117.67
8	1802240	30.69	123.74
16	3604480	68.62	128.25

3. Efficiency of resampling

Our training examples come from a resampling stage. This stage is composed of drawing C candidate examples from the uniform random distribution and resampling K examples from the candidate set. The total time cost can be expressed as

$$T = Ct_1 + Kt_2, \tag{1}$$

where t_1 is the time cost of drawing candidate examples and evaluating weights, and t_2 is the time of drawing examples from candidate set. In practice, resampling an example from a discrete distribution can be conducted in constant time. Therefore, the total time cost T mainly depends on C and K. Commonly, C should be large enough to better represent the underlying target distribution and minimize the bias introduced by that only finite C is used in practice.

In Table 1, we present timings of the two steps for the COUNTRY KITCHEN scene. In this experiment, K is set to $160 \times 88 \times 16$. As can be seen, drawing candidate examples has linear complexity in terms of the candidate sample set size C. C is set to α times of K (i.e., $C = \alpha K$). Figure 6 shows training and test losses of neural networks trained on data from different choices of α. When using a small α, candidate examples are not sufficient to represent the target distribution and neural networks suffer from overfitting problems. Meanwhile, a large α leads to a large candidate set and longer pre-processing time. Taking into account the balance between performance and efficiency, a trade-off choice of α is $4 \sim 8$.

4. Importance sampling high dimensional PSS

In this section, we investigate the ability of neural networks to learn high dimensional distributions in PSS. Specifically, we warp the first 12 dimensions of a PSS vector. To learn this 12D PSS distribution, we use a larger neural network with 16 coupling units, whose total number of internal layers reaches 66 (four internal layers per coupling unit, one input layer and one output layer). For an internal layer within a coupling unit, the number of neurons is set to 60.

Since the neural network has a relatively deep architecture, training datasets should be sufficiently large to avoid overfitting. Figure 7a, shows convergence behaviors of the neural network training under different sizes of training datasets. We evaluate four training datasets with increasing sizes: epp-64, epp-256, epp-1024 and epp-4096. The ratio α of candidate sets is set to 6. From the gap

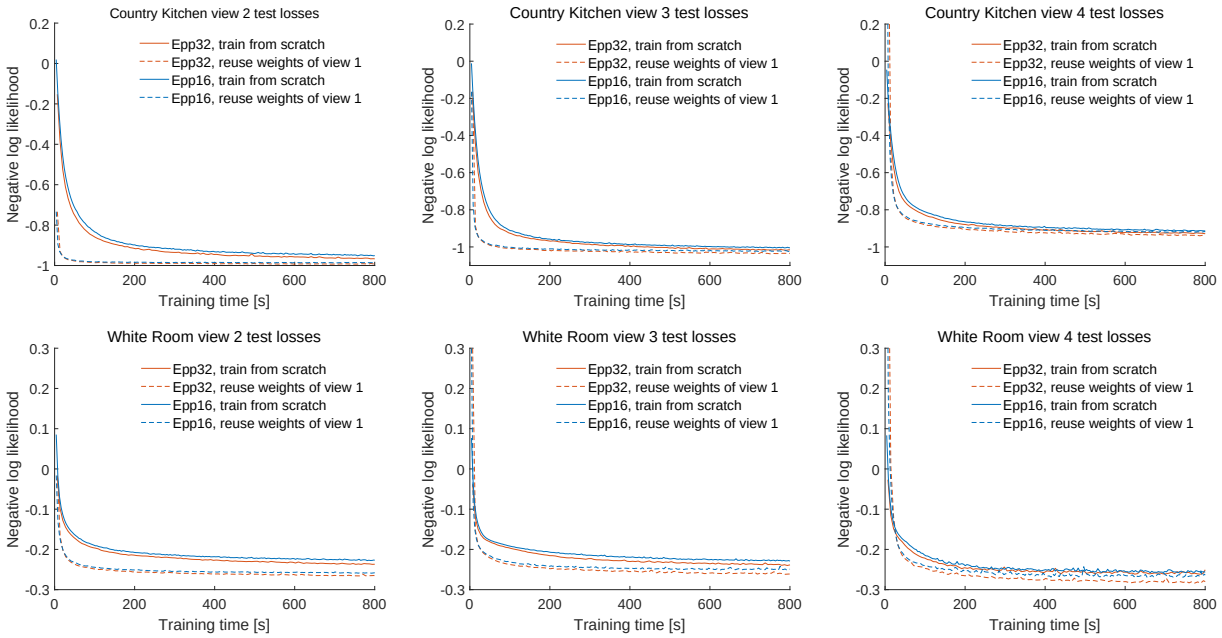

Figure 1: *Comparisons of test losses of the weight reusing approach and training from scratch for 4D PSS warps. Each training session runs for the same time of 800 seconds. View 1 denotes the original camera view. The reusing approach leverages the weights of the neural networks trained on view 1 to initialize the neural networks of the other camera views.*

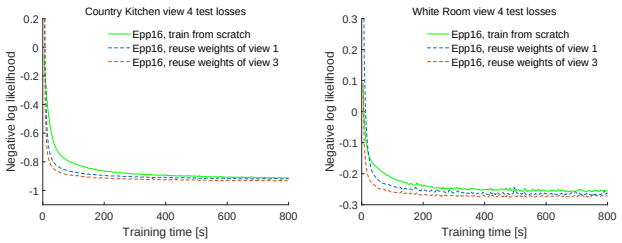

Figure 2: *Test losses of incremental weight reuse and training from scratch. Each training session runs for 800 seconds. The reusing approach leverages previously trained networks of view 1 and 3 to initialize the networks of view 4. Initializing with pretrained weights of a closer view (view 3) further improves training convergence compared to more distant views (view 1).*

between training and test losses it can be observed that the training suffers from overfitting when the number of training examples is too small. As more training examples are used, the gap narrows and the overfitting problem is alleviated. This is, however, accompanied by longer preprocessing time and much longer training time (Table 2).

Figure 8 shows an equal sample count (128 spp) comparison of rendering results. Close-up regions are mainly lit by indirect illumination, which is produced by light paths with many bounces. The neural network training of epp-64 are stopped early at epoch 20, whereas other neural networks are trained for 120 epochs. We observe that neural networks trained on larger datasets lead to lower

Table 2: *Time costs of resampling and training for learning a 12D PSS distribution. The resampling process is implemented with 1 CPU thread. Neural networks are trained for 120 epochs with varying sizes of training datasets. Training example count is equal to epp times of the image resolution* 160×88. *The data come from the* COUNTRY KITCHEN *scene.*

Datasets	Resampling (h)	Training (h)
epp64	0.29	0.84
epp256	0.78	1.93
epp1024	1.76	7.34
epp4096	8.14	25.59

Table 3: *Time costs of resampling and training for learning a 6D PSS distribution. Other settings are the same as in Table 2. The data are from the* COUNTRY KITCHEN *scene.*

Datasets	Resampling (h)	Training (h)
epp4	0.010	0.05
epp64	0.018	0.34
epp256	0.656	1.29
epp1024	2.720	5.19

rendering errors. When the training dataset is small, the trained network is plagued by overfitting, and gives rise to less error reductions.

We further investigate the relation between the training data size and overfitting by learning a 6D PSS distribution from the same

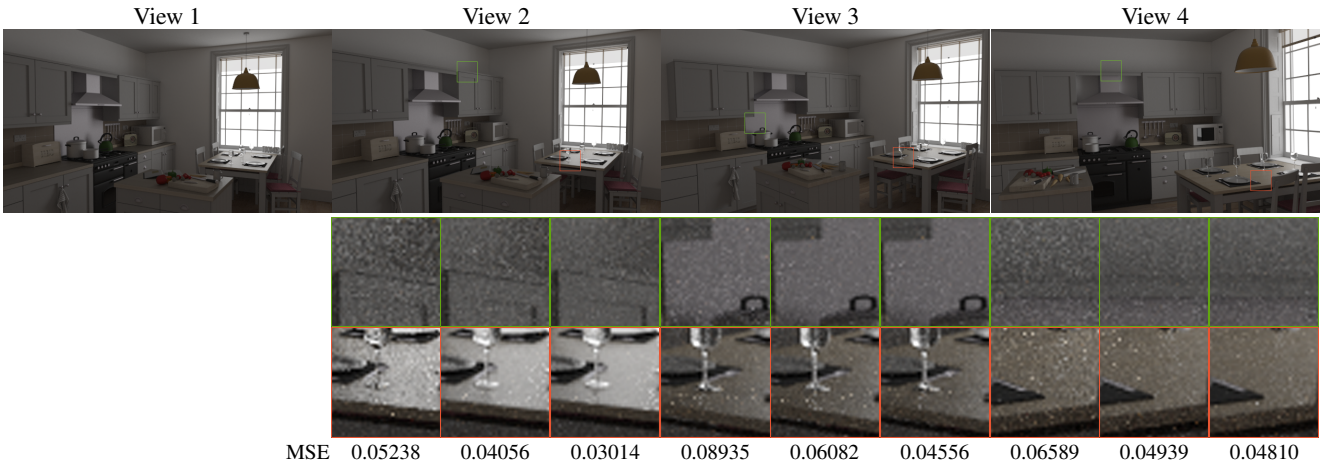

Figure 3: *Network weight reusing for the* COUNTRY KITCHEN *scene. View 1 is the original camera view, whereas the following three views correspond to gaze angle changes of approximately 10, 45, and 90 degrees, respectively. In the close-ups under each reference image, we compare images of the path tracing baseline (left), training from scratch (middle) and networks reusing (right) at equal training time. The preset training time of view 2/3/4 is 200/320/480 seconds.*

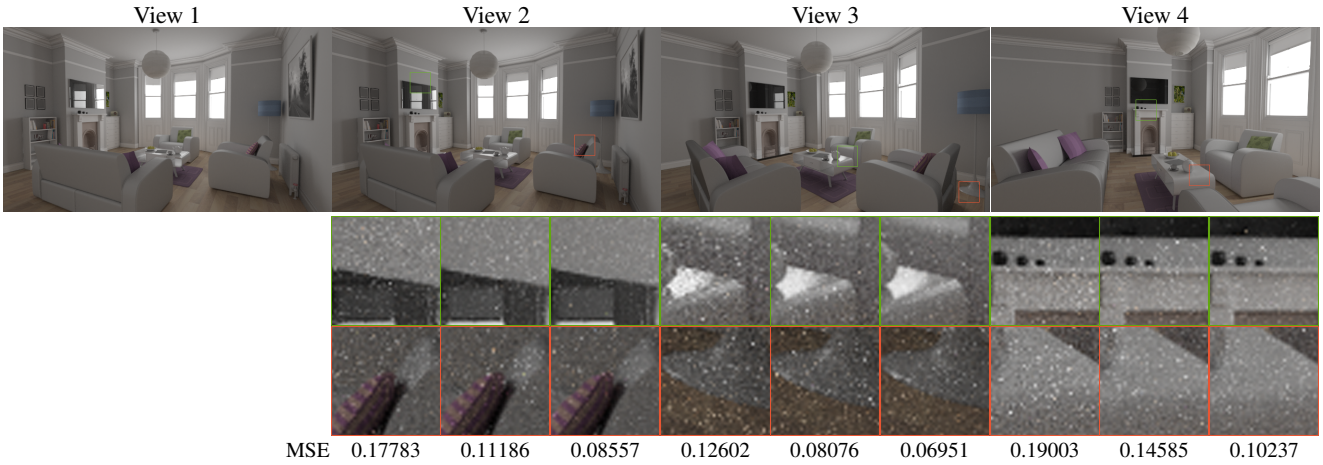

Figure 4: *Network reusing for the* WHITE ROOM *scene. View 1 shows the original camera view, and the other three views correspond to gaze angle changes of approximate 10, 45, and 90 degrees. Close-ups below each reference image are path tracing baseline (left), training from scratch (middle) and network reusing (right). The preset training time of view 2/3/4 is 160/240/400 seconds.*

scene, as shown in Figure 9. Neural networks have the same architecture as before, and they are trained on epp-4, epp-64, epp-256 and epp-1024 training datasets, respectively. We tabulate the time cost of resampling and training in Table 3, and convergence behaviors of neural networks are plotted in Figure 7b. An epp-4 dataset is used in this experiment, and the neural network overfits to the small dataset. As shown in the red close-up of epp-4, there is a change of the noise pattern transitioning from left to right, which is mainly caused by overfitting. Neural network sampling focuses more on the PSS distribution corresponding to the left part and leads to smoother result there. With more training data used, the overfitting issue is mitigated, and the neural sampling effectively reduces the rendering errors.

While the behavior of the trained networks in 6D and 12D are overall quite similar, it can be noticed that the learned 6D PSS warp provides slightly lower rendering errors than the 12D warp. This is related to the "curse of dimensionality" problem. Suppose λ samples are desired to sample each dimension, which will lead to a requirement of λ^{12} examples. The required number of samples grows exponentially as the dimension λ increases, thus 12D learning in theory requires much more data than 6D learning. To take advantage of such a large training dataset, however, we would likely need an even deeper neural network. This would incur prohibitive preprocessing and training time. The high time cost makes learning higher dimensional PSS distribution not practical at present.

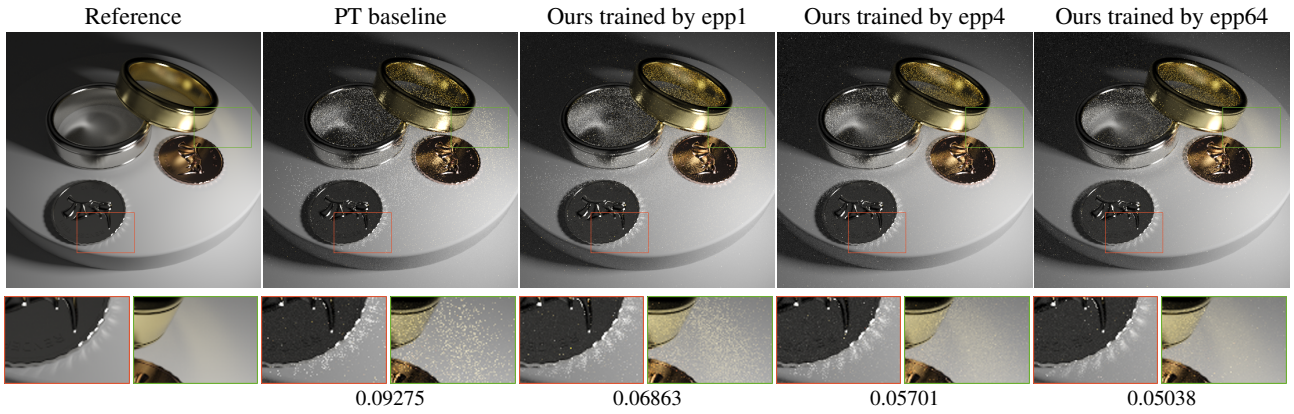

Figure 5: *Comparisons of small illumination features in the* COINS *scene rendered at 128 spp. MSE is given below each image. The reference image is rendered with Metropolis light transport using 32768 spp. Note that our neural approach gives better cardioid caustics in the silver ring, and it preserves small caustics with fine structures.*

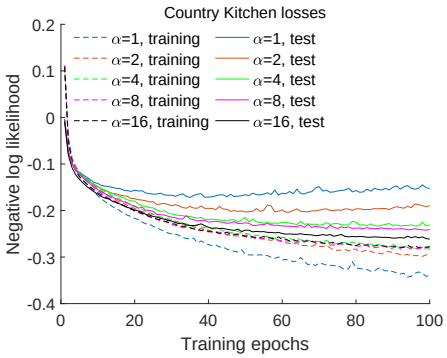

Figure 6: *Test losses of neural networks trained on data produced with different candidate sets. Each training session runs for 100 epochs, and hyperparameters for each session are the same.*

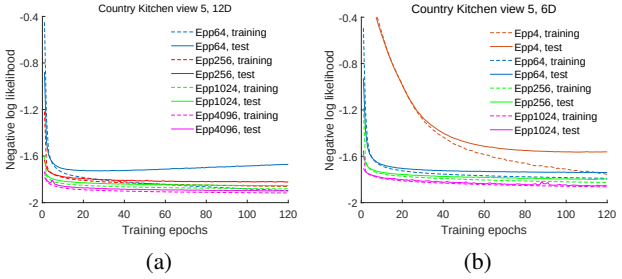

Figure 7: *Test losses of (a) 12D PSS learning and (b) 6D PSS learning, with respect to training datasets of different sizes. Each training session runs for 120 epochs, and hyperparameters for each training session are the same.*

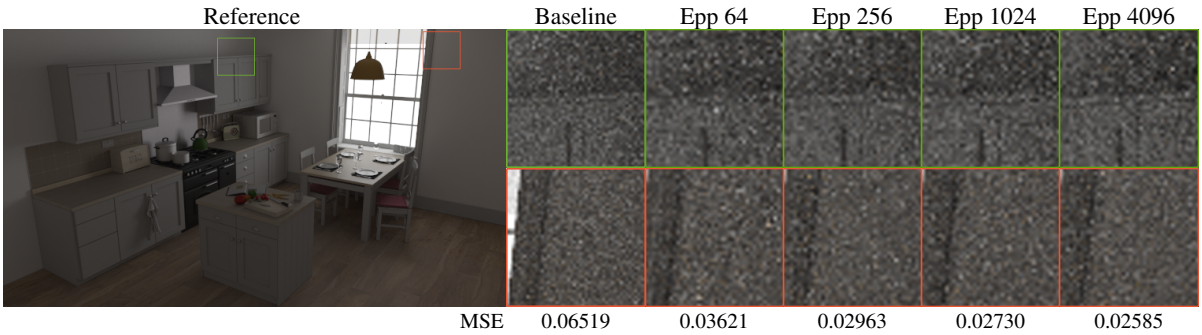

Figure 8: *Equal sample count (128 spp) comparisons of **12D** PSS importance sampling for the* COUNTRY KITCHEN *scene. Close-ups show regions mainly lit by indirection illumination, which depends on light paths with many bounces. Baseline path tracing method uses uniform random PSS sampling. While smaller training datasets lead to some error reduction, our method further improves by using a larger training datasets.*

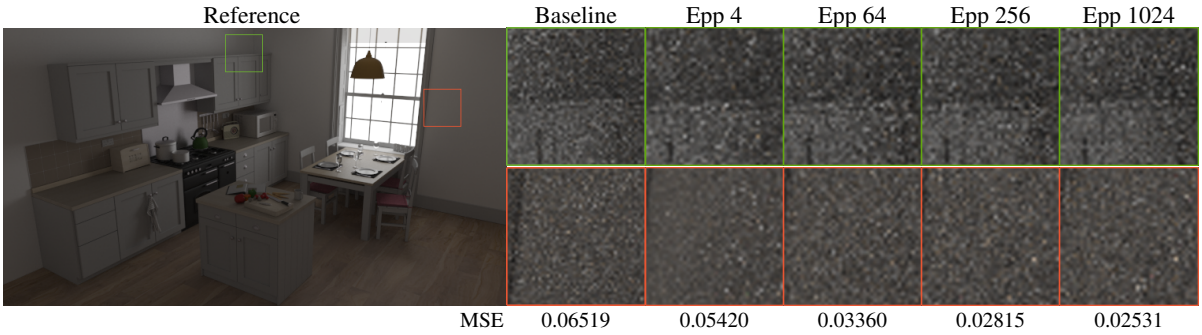

Figure 9: *Equal sample count (128 spp) comparisons of rendering results from **6D** PSS importance sampling. Neural networks are trained on datasets with four different sizes. MSE errors are provided below images. The reference image is rendered by Metropolis light transport using 32768 spp. Similar as in the 12D case, larger training datasets lead to better results.*