Guided-Generative Network for noise detection in Monte-Carlo rendering

Jérôme Buisine, Fabien Teytaud, Samuel Delepoulle, Christophe Renaud

Guided-Generative Network for noise detection in Monte-Carlo rendering

Jérôme Buisine
Univ. Littoral Côte d'Opale
LISIC, Calais, France
jerome.buisine@univ-littoral.fr

Fabien Teytaud
Univ. Littoral Côte d'Opale
LISIC, Calais, France
fabien.teytaud@univ-littoral.fr

Samuel Delepoulle
Univ. Littoral Côte d'Opale
LISIC, Calais, France
samuel.delepoulle@univ-littoral.fr

Christophe Renaud
Univ. Littoral Côte d'Opale
LISIC, Calais, France
christophe.renaud@univ-littoral.fr

***Abstract*—Estimating the features to be extracted from an image for classification tasks are sometimes difficult, especially if images are related to a particular kind of noise. The aim of this paper is to propose a neural network architecture named Guided-Generative Network (GGN) to extract refined information that allows to correctly quantify the noise present in a sliding window of images. GNN tends to find the desired features to address such a problem in order to emit a detection criterion of this noise. The proposed GGN is applied on photorealistic images which are rendered by Monte-Carlo methods by evaluating a large number of samples per pixel. An insufficient number of samples per pixel tends to result in residual noise which is very noticeable to humans. This noise can be reduced by increasing the number of samples, as proven by Monte-Carlo theory, but this involves considerable computational time. Finding the right number of samples needed for human observers to perceive no noise is still an open problem. The results obtained show that GGN can correctly solve the problem without prior knowledge of the noise while being competitive with existing methods.**

***Index Terms*—Deep Learning, GAN, Monte-Carlo, Computer Graphics, Noise detection**

I. Introduction

Modern realistic image algorithms mimics the natural process of acquiring pictures by simulating the physical interactions of light between every existing objects, lights and cameras lying within a 3D modelled scene. Light simulation process in a 3D scene is known as global illumination and was formalised by Kajiya [1] with the light transport rendering equation.

This equation cannot be analytically solved in most cases and Monte-Carlo (MC) approaches are generally used to estimate the value of the final image pixels. Sampling is achieved by constructing random light paths between the camera and the light sources located in the 3D scene in order to collect the contribution of the light emitted by each light source to the pixels in the image.

This research was funded by ANR support: project ANR-17-CE38-0009. Experiments presented in this paper were carried out using the CALCULCO computing platform, supported by SCoSI/ULCO (Service COmmun du Système d'Information de l'Université du Littoral Côte d'Opale).

(a) 1 sample (b) 20 samples (c) $10,000$ samples

Fig. 1: A view of the scene *Kitchen* available from [2] rendered using the PBRT engine [3] with 3 levels of sampling per pixel. The number of samples required can be very high before residual noise is no longer noticeable.

The final MC estimator approximation of the expected value for n samples is obtained from the empirical mean. This computation initially causes considerable noise when generating the image, but as the calculation progresses, this noise is reduced and almost invisible (see Fig. 1).

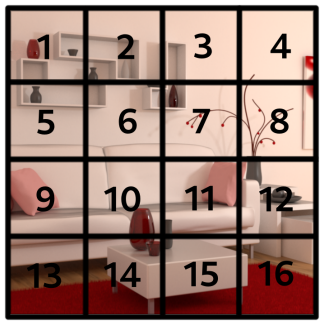
Unfortunately convergence requires often several hours (even days) before a visually usable image is available, due to both the complexity of paths computation and the high number of samples that are required. Then, information about the number of paths that are really required for the image to be visually converged is unknown. Stopping computation too soon provides images with visual noise and computing too much samples can lead to a loss of time and higher production costs. Furthermore the identification of features relevant for the identification of visual noise is difficult given the nature of the noise generated, which is very dependent on the computational algorithms used and the complexity of virtual scenes (material properties, caustic effects, indirect lighting...).

In this paper we propose to exploit Deep Learning methods such as U-Net Denoising Autoencoder [4] and Generative Neural Networks (GAN) [5] to automatically generate Noise Feature Maps (NFM) which guide a Discriminator neural network to better characterize the task of identifying humanly perceptible noise in images.

Based on these ideas, the work presented in this paper will be organised as follows: first, in Section II the previous works related to this problem are presented. Then, in Section III the proposed neural network architecture is introduced. The results obtained on a large database are compared to previous methods in Section IV before concluding on the performance of the developed method and the perspectives of this work.

II. Previous works

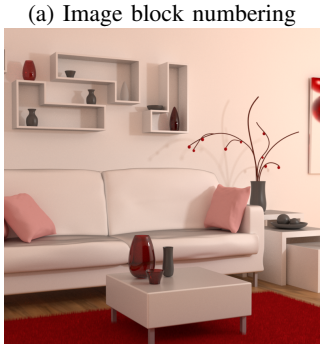
When trying to identify noise in an image, two problems arise; on the one hand, it is necessary to find attributes to characterise this noise and on the other hand, it is necessary to have access to the visual detection thresholds of this artefact for many images in order either to train learning models or to verify the results obtained. On this last point, the works carried out in [6] and [7] use a base of 12 images calculated with different sampling levels. These images, of size 512×512, have been cut into 16 disjoint blocks (of size 128×128) for which noise prediction thresholds have been acquired from human users. However, these images are not publicly available and the visual effects that appear in them do not cover all the effects that exist in real images. More recently a larger database has been published in [8] for which 40 images with various light effects are available with a very large number of sampling levels and associated human perceptual thresholds. The images are of size 800×800 and are sliced in a manner similar to previous work into 16 blocks of size 200×200. As this database will be used in our works, Fig. 2 illustrates the way in which the data relating to these images are presented: division into blocks, human perceptual thresholds associated with each block, reference image calculated with a very large number of samples.



(a) Image block numbering

8680	5286	6553	2293
7153	7086	6420	2626
4420	6580	4560	7820
7086	8286	6286	6693

(b) Human thresholds



(c) Reference image

(d) Reconstructed

Fig. 2: Some of the data associated to the image dataset available in [8] : (a) the 16 image blocks, (b) the human visual threshold for each block (in samples per pixel) (c) the reference image computed with $10,000$ samples per pixel and (d) the image reconstructed from the human thresholds.

Identifying the representative characteristics of MC noise is also a complex task and various proposals have been made. [6] proposed to extract 26 features from each image block. They first extract the L channel from Lab color space and then apply four different denoising algorithms to this channel: linear filtering with averaging filters of sizes 3×3 and 5×5, linear filtering with Gaussian filters of same sizes and with standard deviations $\sigma \in \{0.5, 1, 1.5\}$, median filters and adaptive Wiener filters of same sizes. From these 13 denoised gray images, they compute the mean and standard-deviation in order to get their 26 features. These features are then used as inputs of a *Support Vector Machine* (SVM) classifier that allows to predict the binary label (noisy, noiseless) of each image block. In [7], authors proposed to first compute an approximated reference image using quick ray-tracing technique [9] which is computed once at the beginning of the image rendering process. Next, they compute a noise mask for both the current image obtained during the rendering process and the reference image. These masks are computed by applying a Gaussian filter to their respective image before obtaining the absolute difference between the initial image and the filtered image. Finally, the two masks are passed to an SVM model that allows to classify the current image block as still noisy or not where each block is of size 128×128. Giving 2 images directly as input to an SVM model can lead to what is called the dimension curse problem, as the image size increases. The SVM model will therefore need more time to learn and find its hyper separator plane.

More recently [10] proposed to use the SVD-Entropy [11] measurement as a noise feature. They compute the SVD-Entropy on each block, taking into account a sliding window of size S for which each version of the block has a decreasing noise level. A sub-part of the SVD-entropy vector is extracted for each block of the window and used for training a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) cells [12]. The obtained model seems to generalize correctly on the unlearned blocks of some images but it seems that it does not work in some particular cases.

III. Guided-Generative Network

As mentioned earlier, it is difficult to identify the features that are representative of noise in images generated in MC methods. In this paper, we aim to propose an approach where these features are generated automatically before binary classification. For this purpose, we rely on the notion of noise mask [7] which will be provided by a generative neural network model and the use of a sliding window of images that appears to bring a better robustness of the models [10]. The proposal for such an approach is justified by the emergence

of robust deep convolution learning methods for both noise processing and recognition [13, 14].

Our proposed GGN architecture whose purpose is to obtain automatic noise-related features is composed of 3 neural network models, for which a sliding window of images of different noise levels of size S is used. Before going into the details of each sub-model, Fig. 3 illustrates the desired interaction between each of them.

A. Denoising Autoencoder

In [7], a noise mask is computed from both the image being calculated (potentially noisy) and an approximate reference image, obtained by the ray-tracing method. The main drawback is that some important light effects cannot be simulated by ray tracing and thus induces errors as compared to the final image that should be computed. The idea proposed to remedy this problem is an Autoencoder neural network to best denoise an image. In Computer Graphics, the preservation of structures and light effects is important, that's why the Autoencoder used is of type U-Net as exploited for denoising task in [4, 15], since it makes it possible to preserve the structure of the image more easily while proposing a powerful denoising of the input image. U-Net is a convolutional autoencoder with skip connections and regular dimensionality progression. The proposed encoder and decoder have symmetrical structure: each 4 encoder stages uses two 3×3 convolutions and doubles the dimensional depth K, while each 4 decoder stages has two 3×3 deconvolutions and reduces the depth by half. All intermediate stages use batch normalization and LeakyReLU activation functions. The output stage has two 3×3 deconvolutions with LeakyReLU, and a final 1×1 convolution with LeakyReLU activation to output the final denoised image. Each downsampling stage uses a 2×2 MaxPooling, and the upsampling stages use 2×2 Bilinear Upsampling. An additional ZeroPadding layer is used for the decoder stage when it is necessary to obtain an odd tile size after upsampling. We set $K = 16$ as image input is of size 200×200 which implies a rather substantial storage. Also, the Structural Similarity metric (SSIM) [16] was used for loss function i.e., $L(\hat{y}, y) = 1 - SSIM(\hat{y}, y)$ where $\hat{y}$ is the known reference block image computed with $10,000$ samples, and y is the output image of the U-Net neural network. The SSIM also offers good structure preservation [17, 18] rather than a function of L1 or L2, both related to pixel values. As mentioned before, the use of a sliding window of input images of size S is actually processed for better detection robustness later on. Each image of different noise level is sent one after the other to the U-Net in order to be denoised and to obtain a sliding window of S approximated reference images. Thus, the model learns to denoise the images on several noise levels.

B. Feature Map Generator

The Feature Map Generator is also an autoencoder and has the same structure as the previous denoising model but without skip connections and regular dimensionality progression as proposed by a U-Net model. Then a final 1×1 convolution is applied with LeakyReLU activation to output the final expected gray image with $K = 1$. Hence, this model takes as input an image sliding window of size S and aims at producing a NFM with unknown expectation. In our approach and as detailed in Fig. 3, it will take either the sliding window of input image or the sliding window of approximated reference images obtained previously. It's important to note that models such as the Variational AutoEncoders (VAE) [19] which generally offer better generated data were also tested but did not provide good results.

C. Discriminator for binary classification

Finally, the discriminator takes as input 2 NFM obtained from the Feature Map Generator, either the NFM from the input sliding window or the NFM from the sliding window of approximated reference images. The discriminator must provide a binary label, so it is first composed of 3 convolution layers which doubles the dimensional depth K with a kernel size of 3×3 and padding of 2 to reduce the dimensionality of the NFM. For each of these convolution layers, intermediate layers of batch normalization, LeakyReLU and MaxPooling with a kernel size of 3, a stride of size 2 and a padding of size 1 are processed. Then 4 classical linear layers are exploited to propagate and reduce the information until reaching a probability of belonging to a noisy or non-noisy label. The first linear layer returns data of size $K \times 4$ then the two following layers reduce the output data by half. The last linear layer proposes a single output before applying a Sigmoid function. Each linear layer applies intermediate layers such as a batch normalization, a LeakyReLU layer and a 50% dropout layer to avoid overfitting. Only the last layer, where the probability is obtained, does not have a dropout layer.

The Binary Cross-Entropy loss function [17] is used to both propagate the error of the discriminator, but also the Feature Map Generator. This allows to obtain NFM that are representative of the expected classification task and to guide the discriminator, hence the name Guided-Generative Network for such an architecture.

The next section details the training procedure of such an architecture and the results we obtained.

IV. Results

The proposed architecture composed of the 3 models described above is trained on the synthetic image data where human thresholds are available. The 3 models learn together with respect to the input data as shown in Fig. 3. From the 40 images available in [8], 35 have been selected for training, and 5 for verifying the model performance.

A. Experimental setup

The amount of data is quite significant with 35 images, where each image is composed of 16 blocks of size 200×200. For each image 500 different noise levels ranging from 20 samples to $10,000$ samples are used. This allows to learn from a total of $280,000$ labeled data. Regarding the training parameters, the model was trained on the data for 30 epochs,

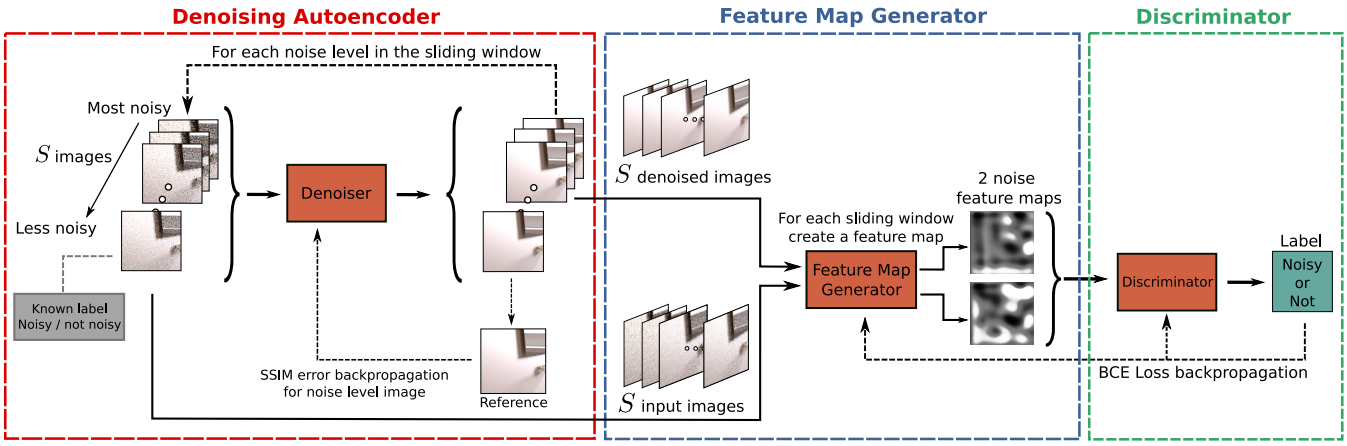


Fig. 3: Proposed model architecture where a sliding window of images of different noise level and of size S is given as input to the Autoencoder model for denoising in order to obtain a sliding window of approximated reference images of size S. One after the other, the two image sliding windows, the input one and the approximated reference one, are sent to the Feature Map Generator model to obtain respectively a noise feature map (NFM). Finally, the two NFM obtained are transmitted to the Discriminator to evaluate whether the last image of the input sliding window is considered to be still noisy or not using. The propagation of the error is then possible thanks to the known label (noisy / not noisy) from the last image of the input sliding window.

with a batch size of 64 and a sliding window of images of size $S = 6$, which we consider sufficiently robust for prediction.

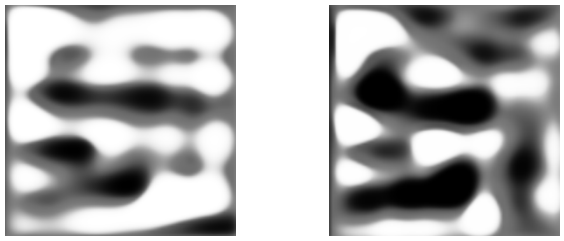
The U-Net Autoencoder model appears to provide MC noise removal results that are close to the reference image, as illustrated in Fig. 4, where the SSIM scores show an improvement in the quality and closeness of the resulting image compared to the reference image.

(a) Noisy image SSIM: 0.6433 (b) Denoised image SSIM: 0.9859 (c) Reference SSIM: 1.0

Fig. 4: Result of the U-Net Autoencoder denoising on one block of the *Kitchen* image with corresponding SSIM scores.

An overview of the outputs of the Feature Map Generator is available in Fig. 5, for the same image block that as been used in Fig. 4. Let us note that differences appear between the two NFMs obtained. These ones are due to the fact that different noise levels coexist in the same sliding window for the images under calculation, whereas these differences are attenuated in the images that are denoised in the second sliding window. This confirms the interest of the approach, as the NFM provides discriminating information as to the presence or absence of noise in the images considered.

These NFM are then sent to the Discriminator for evaluation of the probability of the last image of the input sliding window to belong to label 1, noisy, and label 0, not noisy.



(a) Input noise feature map (b) Approximated reference feature map

Fig. 5: Results of the Feature Map Generator on the block of Fig. 4 : the NFM from the sliding windows of computed images (a) and the NFM from the denoised images (b).

B. Models comparisons

The model obtained after a training of 30 epochs on the training dataset is compared to the RNN proposed by [10] with the same training conditions, i.e. the same train and test dataset, the same batch size and the same sliding window size ($S = 6$). The metrics used to compare the performance of the two models are the accuracy and the area under the ROC curve (AUC ROC) [20], which defines how well the model separates the two classes of the binary classification.

The performance of the models on the training and testing datasets is presented in Table I with several decision thresholds in order to check the accuracy performance. The LSTM model seems to have a better generalisation. However, even if the GGN has a slight overfitting, it shows a correct performance in test when its decision threshold $t \geq 0.95$. In addition, Fig. 6 shows the predictions of the models during the rendering of an image and illustrates the general behaviour of the prediction

	t threshold	Accuracy Train	AUC ROC Train	Accuracy Test	AUC ROC Test	Accuracy Global	AUC ROC Global
LSTM	0.3	0.7619	0.9115	0.8122	0.9297	0.7682	0.9137
	0.4	0.8085	0.9115	0.8456	0.9297	0.8131	0.9137
	0.5	0.8281	0.9115	**0.8519**	0.9297	0.8311	0.9137
	0.6	0.8329	0.9115	0.8385	0.9297	0.8336	0.9137
	0.7	0.8225	0.9115	0.8055	0.9297	0.8204	0.9137
	0.8	0.7965	0.9115	0.7542	0.9297	0.7912	0.9137
	0.9	0.7533	0.9115	0.6896	0.9297	0.7454	0.9137
	0.95	0.7331	0.9115	0.6654	0.9297	0.7246	0.9137
	0.98	0.6888	0.9115	0.6284	0.9297	0.6812	0.9137
GGN	0.3	0.6598	0.9735	0.6641	0.8422	0.6603	0.9571
	0.4	0.7152	0.9735	0.6719	0.8422	0.7098	0.9571
	0.5	0.7902	0.9735	0.7310	0.8422	0.7828	0.9571
	0.6	0.8219	0.9735	0.7415	0.8422	0.8118	0.9571
	0.7	0.8553	0.9735	0.7554	0.8422	0.8429	0.9571
	0.8	0.8809	0.9735	0.7709	0.8422	0.8672	0.9571
	0.9	0.9067	0.9735	0.7858	0.8422	0.8916	0.9571
	0.95	0.9204	0.9735	0.8013	0.8422	0.9055	0.9571
	0.98	0.9201	0.9735	**0.8216**	0.8422	0.9078	0.9571

TABLE I: Performance of the models on the training and testing sets with several proposed **t**-probability decision thresholds for comparing the accuracy of each model. The AUC ROC score therefore does not change for the **t**-value but remains indicative. The best accuracy scores are indicated with a grey background corresponding to the **t**-value.

results obtained. The decision threshold of GGN is set at $t = 0.95$ as it avoids early prediction on the whole set of images in the training database although its accuracy is lower. Indeed, a conservative capability, although less effective, is to be preferred to an early termination, which would imply a poor quality of the final image obtained. It can be noticed that GGN fluctuates and hesitates much less than LSTM which indicates a stability of the model when predicting. It allows in particular in certain cases to avoid a prediction too early of computation as illustrated for a block of the *Classroom* image. On the other hand, GGN tends at times to predict later than LSTM such as for the image block *San-Miguel*. GGN shows good results even though initially no indication of noise features was given. The model slightly overfits but provides predictions that are fairly accurate.

V. Conclusion

In this paper we propose a GGN architecture that allows to generate accurate noise detection data to guide the binary classification task. In this way, it is not necessary to worry about using a precise method for extracting such features. The results of the proposed GGN model suggest that the automated generation of such features as input to a classification model relative to the expected task is relevant. Compared to other more recent approaches, the GGN brings equally significant or even more precise results on the way to predict. Indeed, its decision threshold is more robust and conservative, which is a preferable behaviour in Computer Graphics. In fact, stopping too early leads to a remaining perceptible noise, whereas stopping later allows the quality to be identical (or improved) even if the computation time is more substantial.

The proposed method has been tested here on noise induced by a Monte-Carlo path tracing algorithm. Given the generality of the GGN, the same method could be adapted to other kind of visual artifacts caused by other computational methods.

One of the perspectives considered in relation to the work done is to study methods to avoid such model to overfit over training data. We then plan to study the use of smaller image blocks (typically 50×50 pixels), to enable us to check whether a bias is not introduced during the training process on larger blocks. The noise may indeed not be uniform in an block of size 200×200, complicating the task of the learning algorithm. This will however require to measure some more accurate human thresholds for noise perception, that were not available in [8]. Another perspective would be to use this same type of approach but for stereoscopic images where the human thresholds would be captured relative to two images (left image and right image) in order to verify the robustness of such an approach.

References

[1] J. T. Kajiya, "The rendering equation," in Proceedings of the 13th annual conference on Computer graphics and interactive techniques, 1986, pp. 143–150.

[2] B. Bitterli, "Rendering resources," 2016, https://benedikt-bitterli.me/resources/.

[3] M. Pharr, W. Jakob, and G. Humphreys, Physically based rendering: From theory to implementation. Morgan Kaufmann, 2016.

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, ser. Lecture Notes in Computer Science, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Springer International Publishing, 2015, pp. 234–241.

[5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014. [Online]. Available: http://arxiv.org/abs/1406.2661

[6] J. Constantin, A. Bigand, I. Constantin, and D. Hamad, "Image noise detection in global illumination methods based on FRVM," vol. 164, pp. 82–95, 2015. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0925231215002957

[7] N. Takouachet, S. Delepoulle, C. Renaud, N. Zoghlami, and J. M. R. Tavares, "Perception of noise and global illumination: Toward an automatic stopping criterion based on SVM," vol. 69, pp. 49–58, 2017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0097849317301607

[8] J. Buisine, S. Delepoulle, R. Synave, and C. Renaud, "Subjective human thresholds over computer generated images," Feb. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4964303

[9] T. Whitted, "An improved illumination model for shaded display," in Proceedings of the 6th annual conference on Computer graphics and interactive techniques, 1979, p. 14.

[10] J. Buisine, A. Bigand, R. Synave, S. Delepoulle, and C. Renaud, "Stopping criterion during rendering of

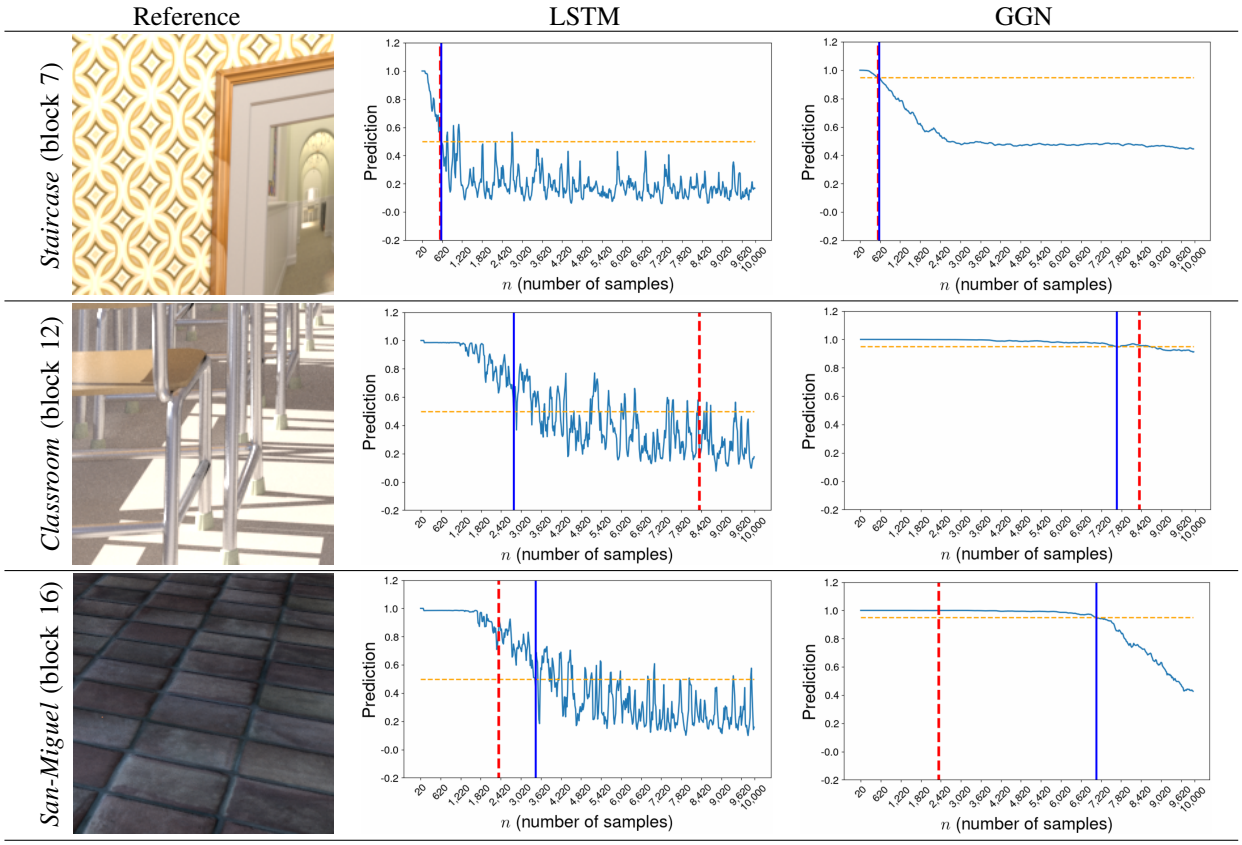


Fig. 6: Comparison of LSTM and GGN model predictions on some blocks of 3 images from the test set for each noise level ranging from 20 to $10,000$ samples. The vertical dashed red curve represents the expected human threshold, the horizontal dashed orange curve represents the model prediction threshold, and the vertical blue curve represents the model noise threshold prediction once the probability decision threshold is reached. The decision threshold $\mathbf{t}$ for the LSTM model is set at 0.5 and that of the GGN is set at 0.95 which makes it possible to avoid predicting the end of the block computation too early.

computer-generated images based on SVD-entropy," vol. 23, no. 1, p. 75, 2021. [Online]. Available: https://www.mdpi.com/1099-4300/23/1/75

[11] M. Banerjee and N. R. Pal, "Feature selection with SVD entropy: Some modification and extension," vol. 264, pp. 118–134, 2014.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[13] M. Haselmann, D. P. Gruber, and P. Tabatabai, "Anomaly detection using deep learning based image completion," in 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), 2018, pp. 1237–1242.

[14] C. Pravin and V. Ojha, "A novel ecg signal denoising filter selection algorithm based on conventional neural networks," in 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA), 2020, pp. 1094–1100.

[15] G. Jiang and B. Kainz, "Deep radiance caching: Convolutional autoencoders deeper in ray tracing," 2019, version: 1. [Online]. Available: http://arxiv.org/abs/1910.02480

[16] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," vol. 13, no. 4, pp. 600–612, 2004. [Online]. Available: http://ieeexplore.ieee.org/document/1284395/

[17] A. Buja, W. Stuetzle, and Y. Shen, "Loss functions for binary class probability estimation and classification: Structure and applications," Working draft, November, vol. 3, 2005.

[18] H. Shi, L. W. 0003, W. Tang, N. Zheng, and G. Hua, "Loss functions for person image generation." in BMVC, 2020.

[19] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2014. [Online]. Available: http://arxiv.org/abs/1312.6114

[20] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," vol. 30, no. 7, pp. 1145–1159, 1997. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320396001422