Guided-Generative Network: A New Robust Deep Learning Architecture for Noise Characterization in Monte-Carlo Rendering

Jérôme Buisine, Fabien Teytaud, Samuel Delepoulle, Christophe Renaud

Guided-Generative Network: A new robust deep learning architecture for noise characterization in Monte-Carlo rendering

Jérôme Buisine, Fabien Teytaud, Samuel Delepoulle and Christophe Renaud

Abstract Current methods for generating realistic computer-generated images rely on stochastic lighting simulation techniques based on a Monte Carlo approach. These Monte Carlo simulations construct light paths between the camera and light sources within the virtual 3D model to calculate the appearance of objects and provide realistic images. Insufficient sampling of the light path space results in high variance between individual pixels in the image which is visually perceived as noise. To reduce this noise, the number of light path samples must be increased until the noise is no longer visible, but this has the disadvantage of significantly increasing the computation time. Finding the right number of samples needed for human observers to perceive no noise remains an open problem that this paper addresses using a new neural network architecture called Guided-Generative Network (GGN). As it is often difficult to extract features from an image for classification tasks, the GGN attempts to automatically find the desired features for noise detection. This is done through an architecture composed of 3 models that collaborate to characterize the noise present and guide the classification. The results obtained show that the GGN can correctly solve the problem without prior knowledge of the noise while being competitive with existing methods. A visual validation experiment of the images obtained by the model indicates a significant reliability to the requested task.

1 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

Jérôme Buisine, Fabien Teytaud, Samuel Delepoulle and Christophe Renaud
Univ. Littoral Côte d'Opale, LISIC, Calais, France, e-mail: {jerome.buisine,fabien.teytaud,samuel.delepoulle,christophe.renaud}@univ-littoral.fr

scene is known as global illumination and was formalised by Kajiya [14] with the light transport rendering equation 1:

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_\Omega L_i(x, \omega_i) \cdot f_r(x, \omega_i \rightarrow \omega_o) \cdot \cos\theta_i d\omega_i \qquad (1)$$

where (see Fig. 1 for the annotations):

- $L_o(x, \omega_o)$ is the lightness traveling from point x in direction ω_o;
- $L_e(x, \omega_o)$ is point x emitted lightness (it is null if point x does not lie on a light source surface);
- the integral represents the set of lightness L_i incident in x from the hemisphere of the directions Ω and reflected in the direction ω_o. The reflected lightness are weighted by the materials' reflecting properties (bidirectionnal reflectance function $f_r(x, \omega_i \rightarrow \omega_o)$) and the incident cosine angle.

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.

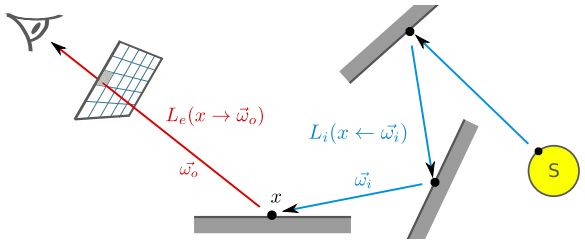


Fig. 1: A generated light path reaching a light source present in the scene where the term L_o is obtained by the sum of the terms L_e and L_i.

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 computation due to insufficient sampling of the path space, but as the number of samples progresses, this noise is reduced and almost invisible (see Fig. 2).

As a matter of fact, 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. Early stopping the computation leads to considerable visual noise in the image. On the other hand, continuing the computation with a high number of samples results in substantial computation costs. Furthermore the identification of features relevant for the identification of visual noise is difficult given the nature of the noise

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

Fig. 2: A view of the scene *Kitchen* available from [2] rendered using the PBRT engine [16] with 3 levels of sampling per pixel. The number of samples required can be significantly high before residual noise is no longer noticeable.

generated, which is awfully dependent on the computational algorithms used and the complexity of virtual scenes (material properties, caustic effects, indirect lighting, etc.).

In this paper, we propose an approach exploiting deep learning methods such as U-Net denoising autoencoder [18] and generative neural networks (GAN) [10] to automatically generate noise feature maps (NFM) which guide a discriminator neural network to better characterize the task of identifying visually perceptible noise in computer generated images.

Based on these ideas, the work presented in this paper is organised as follows: first, in Section 2, the previous works related to this problem are presented. Then, in Section 3, the proposed neural network architecture is introduced. The results obtained on a large database are detailed in Section 4, with comparisons to previous methods. In Section 5, a noise perception experiment is proposed in order to validate the performance of the GGN model on the noise detection task, before concluding and discussing the perspectives in Section 6.

2 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 [8] and [20] 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 [5] for which 40

images with various light effects are available with a 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 non-overlapping blocks of size 200×200. Fig. 3 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 large number of samples.

(a) Image block numbering (b) Human thresholds (c) Reference image (d) Reconstructed

Fig. 3: Some of the data associated to the image dataset available in [5] : (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. [8] proposed to extract 26 features from each image block. Each colour image represented in the RGB colour space is first converted to the Lab colour space. The L channel, which appears to be the most noise-sensitive [7], is selected and 13 denoising filters are applied to it separately to extract the desired features: 2 linear filters with averaging filters of sizes 3×3 and 5×5 ; 6 linear filters with Gaussian filters of same sizes and with standard deviations $\sigma \in \{0.5, 1, 1.5\}$; 2 median filters and 2 adaptive Wiener filters of same sizes. An additional wavelet filter is applied in order to provide a 13^{th} image without high-frequency values. They then compute the mean and standard deviation of the pixel's value from these 13 gray images 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 [20], authors proposed to first compute an approximated reference image using quick ray-tracing technique [23] 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. Noise masks are obtained following an approach used in astrophotography [9]: a Gaussian convolution with a convolution coefficient σ is first applied to the original image, then the mask is obtained by the difference between it and the blurred 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 curse of dimensionality, as the image

size increases. The SVM model will therefore need more time to learn and find its hyper separator plane.

More recently [4] proposed to use the SVD-Entropy [1] measurement as a noise characterization feature. They consider a sliding window composed of S images (see Fig. 4), for which each block composing them has a different and decreasing noise level due to a different progress state in the computational process. Then, they compute the SVD-Entropy over the S noise levels for each block and use a recurrent neural network (RNN) model with long short-term memory (LSTM) cells [12] to classify the last image of the sliding window as still noisy or not.

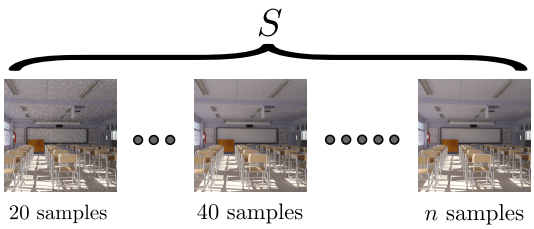


Fig. 4: Representation of a sliding window composed of S images; each image in the window has an increasing number of samples per pixel and therefore a decreasing noise level.

3 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 [20] 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 [4]. The proposal for such an approach is justified by the emergence of robust deep convolution learning methods for both noise processing and recognition [11, 17].

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. 5 illustrates the desired interaction between each of them.

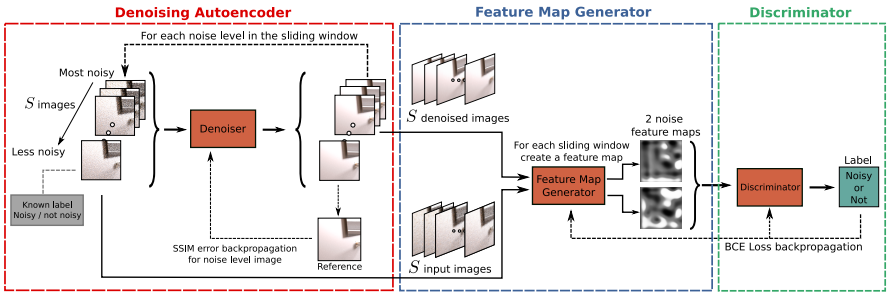


Fig. 5: The proposed model architecture where a sliding window of images of different noise level and of size S is given as input an the autoencoder model. Its task is to denoise the images in order to obtain a sliding window of approximated reference images of size S. One after the other, the two image sliding windows, the input one and the approximated reference one, are sent to the feature map generator model in order to compute their respective noise feature map (NFM) Finally, the two NFMs 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. 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.

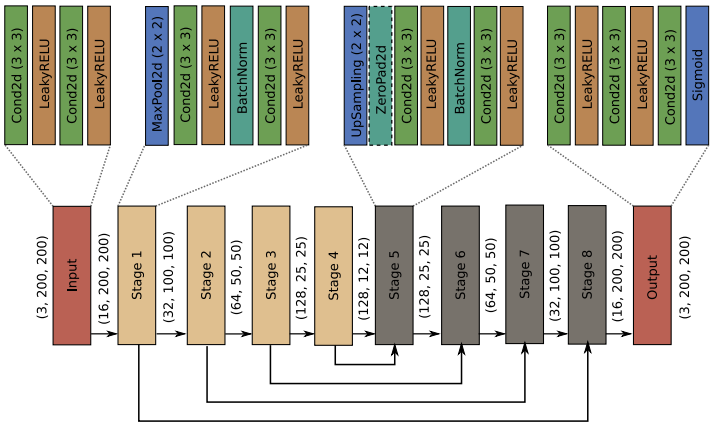


Fig. 6: Representation of the different layers of the U-Net denoising model.

3.1 Denoising Autoencoder

In [20], 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 [18, 13], 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 (see Fig. 6) 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) [21] 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 [6, 19] 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.

3.2 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 (see Fig. 7). 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, as detailed in Fig. 5, it will take either the sliding window of input image or the sliding window of approximated reference images obtained previously. It is important to note that models such as the variational autoencoders (VAE) [15] which generally offer better generated data were also tested but did not provide good results.

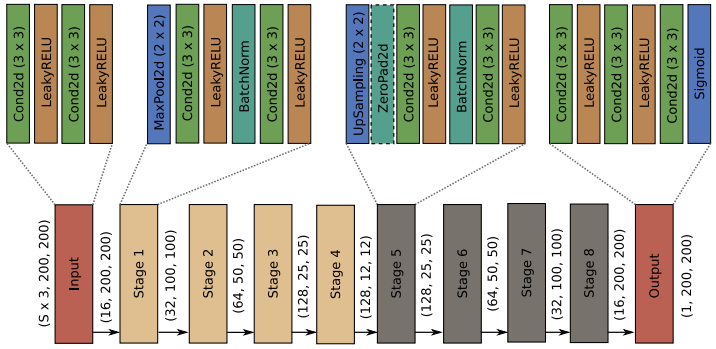


Fig. 7: Representation of the different layers of the Feature Map Generator model.

3.3 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 (see Fig. 8) 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 [6] is used to both propagate the error of the discriminator, but also the feature map generator. This provides representative NFMs of the noise present in the images and guides the discriminator in the expected classification task, 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.

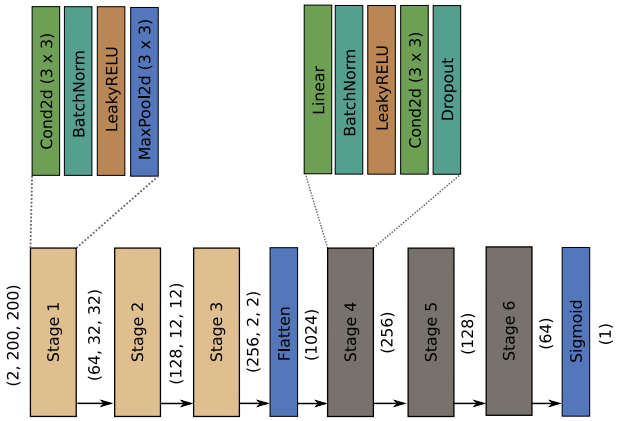


Fig. 8: Representation of the different layers of the discriminator model.

4 Results and comparisons

The proposed architecture is composed of the 3 models. Figures 6, 7 and 8 represent respectively the parameters of the denoiser, the noise feature map generator and the discriminator. The whole code allowing to train the architecture is available at: https://github.com/prise-3d/GGN-MC-rendering. The 3 models are trained together with respect to the input data and the associated human thresholds of the synthetic images (see the proposed training process in Fig. 5). From the 40 images available in the database [5], 35 images have been selected for training, and 5 for verifying the model performance.

4.1 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 to $10,000$ samples per pixel are used. This allows to train the model from a total of $280,000$ labeled data. Input and reference images have been normalized by the maximum value of a colour channel (i.e. 2^8). Regarding the training parameters, the model was trained on the data for 30 epochs, with a batch size of 64 (i.e. 4375 iterations per epoch) and a sliding window of images of size $S = 6$, which we consider sufficiently robust for prediction. The 3 models (denoiser, noise feature map generator and discriminator) use the Adam optimization algorithm.

The experiments were conducted under the Linux system (Debian 10) with a RTX 2080 GPU using the Python programming language (version 3.9) and the PyTorch framework (version 1.10).

4.2 Results overview

The U-Net Autoencoder model appears to provide MC noise removal results that are close to the reference image, as illustrated in Fig. 9, 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

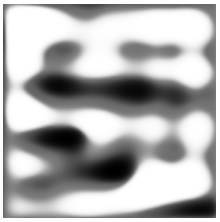
Fig. 9: Result of the U-Net Autoencoder denoising on one block of the *Kitchen* image with corresponding SSIM scores.

An overview of the outputs of the Feature Map Generator is available in Fig. 10, for the same image block that as been used in Fig. 9. 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.

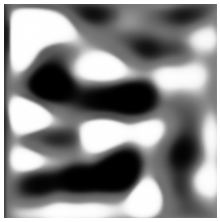
These NFM are then sent to the Discriminator for evaluation of the probability of the last image of the input sliding window to belong to label 1, noisy, and label 0, not noisy.

4.3 Model comparisons

The model obtained after a training of 30 epochs on the training dataset is compared to the RNN with LSTM cells proposed by [4] 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 amount of data exploited from the image database and the input size of the features of the two other methods [8, 20] required as input to the SVM models did not allow us to compare them due to the curse of dimensionality.



(a) Input noise feature map



(b) Approximated reference feature map

Fig. 10: Results of the Feature Map Generator on the block of Fig. 9 : the NFM from the sliding windows of computed images (a) and the NFM from the denoised images (b).

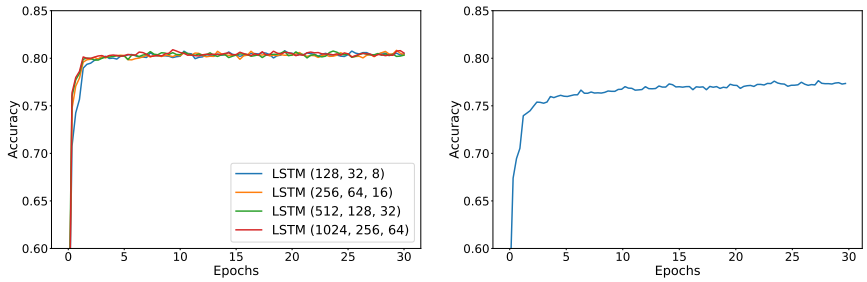
As suggested by [4], RNN model is composed of 3 LSTM layers with the following respective number of hidden units: 512, 128, 32 and a dropout rate of 40% on each one in order to avoid overfitting. Each LSTM layer has a Sigmoid activation function for cell state and hidden state. A Hard Sigmoid activation function is set to activate the input/forget/output gate. A dense layer of size 1 is then used with the Sigmoid activation function as output in order to predict the expected label. As well as the GGN, the LSTM model is trained during 30 epochs where the binary cross-entropy loss is employed to propagate the error during training.

The metrics used to compare the performance of the two models are the accuracy and the area under the ROC curve (AUC ROC) [3], which defines how well the model separates the two classes of the binary classification. Note that the main objective of the GGN model is to be able to predict at least as accurately as the LSTM model while abstaining from finding input features for the classifier, since these features are automatically extracted beforehand.

	LSTM	GGN	
Parameters	**1,604,257**	*Denoiser*	1,380,000
		Generator	1,540,000
		Discriminator	675,650
		Total	**3,595,650**
Training time	52,5 min	6741 min	
FLOPs	**0.000967 G**	*Denoiser*	3.18 G
		Generator	2.32 G
		Discriminator	0.67565 G
		Total	**5.54098 G**

Table 1: Comparison of the complexity of the LSTM and GGN models. The number of trainable parameters of each network, training time and the number of floating-point operations (FLOPs) are indicated in Giga-FLOPs.

Table 1 provides indicators of the complexity of the models. It can be observed that the GGN model has a higher complexity than the LSTM and required a considerable training time. This is mainly due to the fact that the GGN model uses a lot of convolutional layers implying a significant number of floating-point operations (FLOPs) in order to take as input directly the images.



(a) LSTM learning curves with different recurrent layers size

(b) GGN learning curve

Fig. 11: Learning curves of the compared models using accuracy metric. The default decision threshold was fixed to 0.5 in order to compute accuracy.

The learning curves of the two models obtained after 30 epochs are proposed in Fig 11. The accuracy is computed with a decision threshold of 0.5, i.e. the choice of the not noisy label is obtained if the prediction of the model is lower than this threshold (resp. noisy). The parameters of the recurrent layers have been analyzed for the LSTM model by changing the number of hidden units in layers (see sub-figure 11a). Even if we increase the number of parameters, we can observe a certain limit of the performances of the model (the accuracy remains close and still fluctuates). With a decision threshold set to 0.5, GGN model indicates for the chosen decision threshold a lower performance in accuracy than LSTM during the learning process (see sub-figure 11b). In addition to the learning curves, the performance of the models on the training and testing datasets is presented in Table 2 with several decision thresholds in order to check the accuracy performance. The LSTM model seems to have a better generalisation and its performance remains better with a threshold at 0.5. However, even if the GGN has a slight overfitting, it shows a correct performance in test when its decision threshold $t \geq 0.95$ which remains quite close to the LSTM model.

Fig. 12 shows the predictions of the models during the rendering of some images and illustrates the general behaviour of the prediction 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

	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 2: 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.

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. The GGN sometimes tends to predict later than the LSTM, such as with the *San-Miguel* image block, but it is less disturbing than stopping the calculations too early (which may involve a still significant residual noise). Thus GGN highlights good results even though initially no indication of noise features was given. The model slightly overfits but provides predictions that are fairly accurate.

Fig. 13 shows the histogram of the thresholds predicted for each image by each model (histograms in blue). Blocks where users still perceive noise with the maximum number of samples used in our experiments ($10,000$ paths per pixel) are not shown on these graphs, since in this case the human threshold is assumed to be higher. These histograms can be compared to the human thresholds (red histogram). It can be seen that the shape of the GNN histogram seems closer to that of the human thresholds. For the LSTM, its histogram seems more like a Gaussian distribution although flattened. However, the GGN provides many predictions with a value of $10,000$. This corresponds to the fact that it believes that the stopping threshold should be higher than this value, without being able to provide a precise value, since the maximum number of samples in the image database corresponds to this value.

Fig. 14 illustrates the distribution of absolute errors made by each of the two approaches, considering this error as the difference between the human perceptual

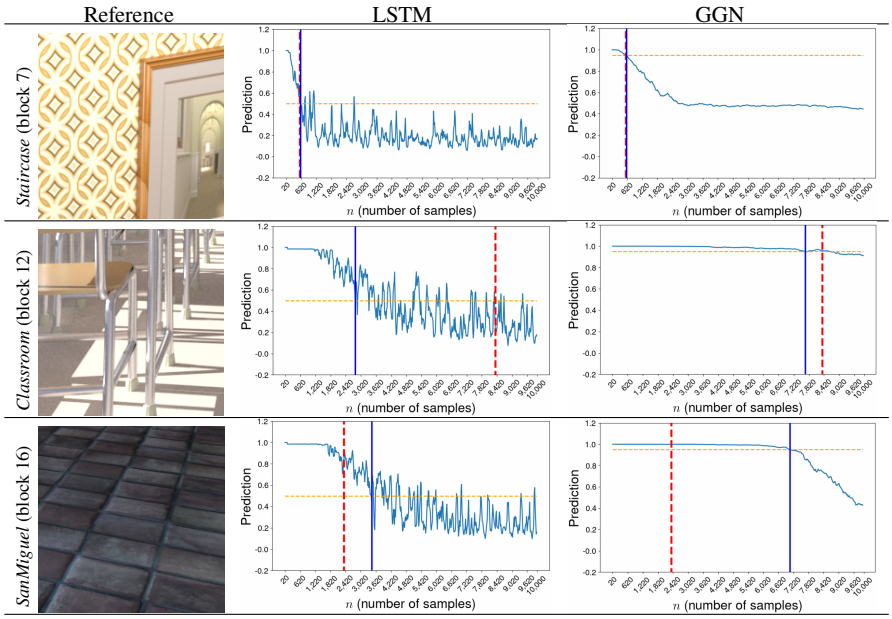


Fig. 12: Comparison of LSTM and GGN model predictions on some blocks of 3 images from the test set for each noise level ranging from 20 to 10, 000 samples. The vertical dashed red curve represents the expected human threshold, the horizontal dashed orange curve represents the model prediction threshold, 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.

threshold and the value predicted by the algorithms for each of the 16 blocks of the 40 images (in this figure all blocks are considered).

These distributions are represented at the same scale in order to better interpret the difference between these two models. The model LSTM with *SVD-Entropy* has a more spread out error distribution, but tends to predict later than earlier. It sometimes stops considerably earlier than the human threshold, resulting in errors of at least 6, 000 on samples (remember that the maximum number of samples in our experiments was set at 10, 000.). On the contrary, the GGN provides thresholds closer to the subjective human thresholds. In addition, it makes fewer early detection errors than its counterpart and the number of overdue thresholds is lower than the LTSM. Note that in this last case the computation time of the corresponding block is longer than that needed to reach the human threshold, but the results obtained do not provide any perceptible noise in the end.

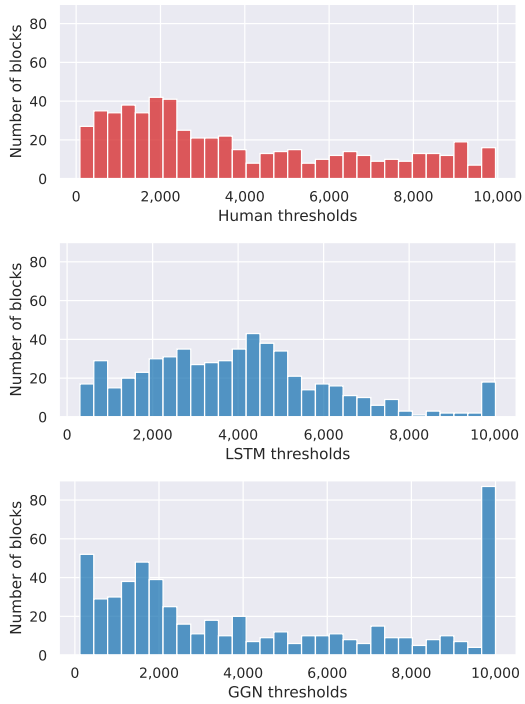
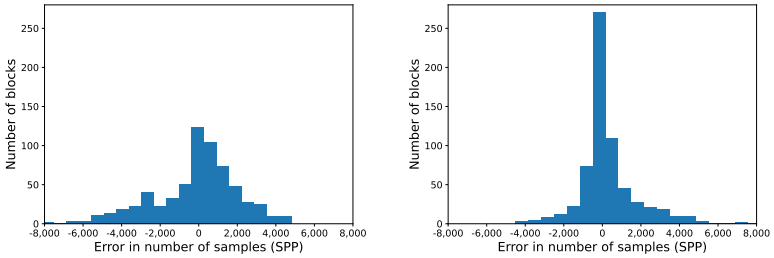


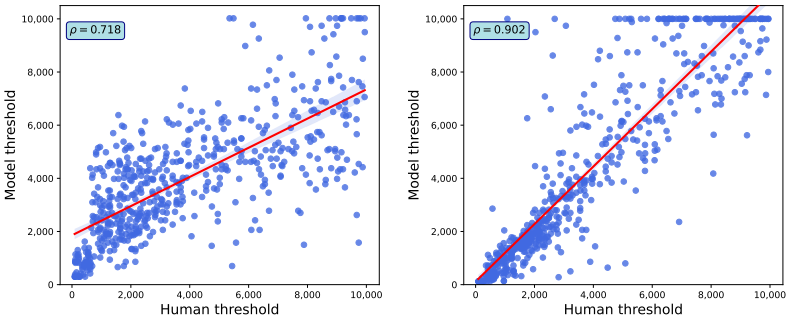
Fig. 13: Histogram of the predicted thresholds over all the blocks of the scenes for each model compared to the histogram of the expected human thresholds. Blocks where human thresholds pointing ahead still noticeable noise on some reference image blocks (i.e. still noisy at $10,000$ samples) are not considered on these graphs.

Fig. 15 further details the predictions of the models against the expected human thresholds on each of the blocks. It allows a clear visualization of the good performances of each model. For both models, only the predictions for human thresholds lower than $10,000$ samples have been presented. We can also note that for some blocks, both models predict that the block is still noisy even after $10,000$ samples (leading to a prediction up to $10,000$ samples). The dispersion of the points is much greater for the LSTM than for the GGN. In fact, the proposed regression line highlights the good performance of the GGN model for the requested task. Moreover, the regression lines indicate that the GGN performs better with a correlation coefficient of 0.902 against 0.718 for the LSTM.



(a) *LSTM* (b) *GGN*

Fig. 14: Distribution of the errors on the LSTM and RNN models computed by the difference of the between produced and subjective thresholds, for each of the 16 blocks of the 40 images.



(a) *LSTM* (b) *GGN*

Fig. 15: Predicted model thresholds versus expected human thresholds for the LSTM and GGN approaches. Some thresholds are not predicted even after 10, 000 samples as the maximum number of samples was set here to 10, 000.

5 Visual experiment

In order to evaluate the performance of the models, a visual experimental validation of the models' predictions has been conducted. Each model predicts a number of samples in a block of 200×200 pixels from which the human should no longer perceive noise. The objective of the validation experiment is to verify whether or not noise is still perceived by a human after reconstructing the images from the models' predictions. A final reconstructed image is obtained in the same way as proposed in Fig. 3, but with thresholds obtained by a model. In this manner, it is possible to ensure the proper behavior or not of the models.

For this purpose, we developed an application, which allows to compare two images, the first one being the image reconstructed from the predicted thresholds of

Fig. 16: Representation of the validation application: the images reconstructed from the predictions of the perception models are displayed next to the images reconstructed from the human thresholds, so that they can be compared by the observers.

a model, the other one being the human reference image, i.e. the one reconstructed from the average human subjective thresholds from the image database (same as proposed in Fig. 3). The question asked to the participants was: *Do the images look the same to you?*. The objective of asking this question to a participant is to know if noise is still perceptually present or not in the image proposed by the model in comparison with the image resulting from the human thresholds.

5.1 Experiment settings and configuration

For some blocks of the 40 images available, noise could still be present on the image considered as reference and computed with $10,000$ samples per pixel. It was not disturbing for the models to have knowledge of this data, because it is the same as having data where noise was always present in the model input. However, for the visual validation of the results, it is necessary that the human reference (reconstructed from the human thresholds) consists of a noise level that is imperceptible to humans.

Therefore, only images with subjective thresholds lower than $10,000$ samples for the 16 blocks were kept for this experiment, i.e. a total of 28 images out of the 40 available in [5].

For each of these 28 images, three images are compared to the reconstructed human image:

- the image reconstructed from the *GGN* model predictions;
- the image reconstructed from the *LSTM* model predictions with *SVD-Entropy* on the same training set of the *GGN* model;

- an image reconstructed from a number of samples eight times lower than the human threshold for each block, in order to have an image in which the noise is visible.

This third noisy image has the objective to let appear noise and to avoid false positives. Indeed, with respect to the question asked, if the participant never notices noise on the reconstructed images from the two models, then he/she risks answering positively to the presence of noise on one of the two, which can lead to the generation of erroneous results (false positives).

The Fig. 16 illustrates the application proposed to the participants, where two images are presented. One of them is the human reference image, the second one is a reconstructed image from a model or with a defined noise level. It is also important to specify that the human image is randomly positioned among the two proposed images, in order to counterbalance a systematic response effect of the position on the right or left of the reference image. The participant can then answer if he perceives a difference between the two images or not. As the human reference is the "ground truth", if no difference is perceptible, this means that the model's performance is correct and meets the expectations.

5.2 Analysis of results

The population of participants for this experiment was quite diverse: students, colleagues, and families, for a total of 17 participants (5 women and 12 men) with an average age of 31. Two subjects with deuteranopia (color blindness) were included in the study, but their data did not seem to differ from the others in retrospect. They reported that their colorblindness did not interfere with their ability to perform the task. The results obtained are presented in the table 3 below which highlights the average percentage of images perceived as noiseless compared to the human reference image for the 3 classes of reconstructed images targeted.

	Noisy	*LSTM*	*GGN*
Images	80/476	403/476	419/476
Percentage	17%	85%	88%

Table 3: Number of images and percentage of images considered not noisy by the users during the experiment for each of the 3 classes of images presented.

In addition to the results obtained and to target the problematic images, the table 4 offers a visual of the results obtained per image (scene) for each of the three

Model	*GGN*		*LSTM*		Noisy	
Scene	Answers	Percentage	Answers	Percentage	Answers	Percentage
Arcsphere	16/17	94.12%	16/17	94.12%	7/17	41.18%
Bunny Fur	16/17	94.12%	16/17	94.12%	3/17	17.65%
Car2	15/17	88.24%	11/17	64.71%	5/17	29.41%
Caustic	14/17	82.35%	13/17	76.47%	2/17	11.76%
Coffee Splash	15/17	88.24%	13/17	76.47%	0/17	0.00%
Crown	14/17	82.35%	15/17	88.24%	4/17	23.53%
Dragon	16/17	94.12%	15/17	88.24%	1/17	5.88%
Ecosys	14/17	82.35%	16/17	94.12%	11/17	64.71%
Eponge Fractal 1	15/17	88.24%	16/17	94.12%	8/17	47.06%
Eponge Fractal 2	14/17	82.35%	13/17	76.47%	2/17	11.76%
Ganesha	17/17	100.00%	15/17	88.24%	7/17	41.18%
Glass Of Water	17/17	100.00%	15/17	88.24%	2/17	11.76%
Indirect	13/17	76.47%	16/17	94.12%	2/17	11.76%
Landscape*	12/17	70.59%	16/17	94.12%	7/17	41.18%
Living Room 2	14/17	82.35%	14/17	82.35%	0/17	0.00%
Living Room 4	16/17	94.12%	16/17	94.12%	0/17	0.00%
Low table	13/17	76.47%	15/17	88.24%	1/17	5.88%
Pavilion Day 1	15/17	88.24%	12/17	70.59%	2/17	11.76%
Pavilion Day 2	14/17	82.35%	12/17	70.59%	1/17	5.88%
Pavilion Day 3	15/17	88.24%	15/17	88.24%	3/17	17.65%
Pavilion Night 2	16/17	94.12%	12/17	70.59%	0/17	0.00%
Sanmiguel 1	14/17	82.35%	15/17	88.24%	3/17	17.65%
Sanmiguel 2*	12/17	70.59%	15/17	88.24%	1/17	5.88%
Staircase 1*	12/17	70.59%	16/17	94.12%	1/17	5.88%
Staircase 2	15/17	88.24%	15/17	88.24%	2/17	11.76%
Tt	16/17	94.12%	15/17	88.24%	0/17	0.00%
Vw Van	16/17	94.12%	10/17	58.82%	1/17	5.88%
Test percentage	36/51	70.05%	47/51	92.15%	-	-

Table 4: Results of the experiment for each image with respect to the number of participants. The * character specifies that the scene has been used in the test set.

reconstructed images in relation to the human reference image. The results on the images from the test set are also detailed for each of the models for which we can note that the *GGN* model is less performing than the *LSTM* model with *SVD-Entropy*, 70.05% and 92.15% respectively. This is mainly due to the fact that a generative model tends to overfit [24].

By analyzing the 3 images from the test set that were included in the 28 selected images, blocks of the images reconstructed *Staircase 1* and *Landscape* by the GGN model do indeed still contain slight noise. However, the case of the image *Sanmiguel view 2* is particular, only one predicted threshold of a block is lower than the human reference.

Figure 17 proposes a representation of the differences of predictions on the 3 problematic scenes by using heat maps. This representation uses a heatmap associated to the number of samples from visual thresholds and predicted by each model: for each block the colors vary from blue to red (from 0 to $10,000$ of samples), in order to be able to visually compare the different results. It highlights the difficulties

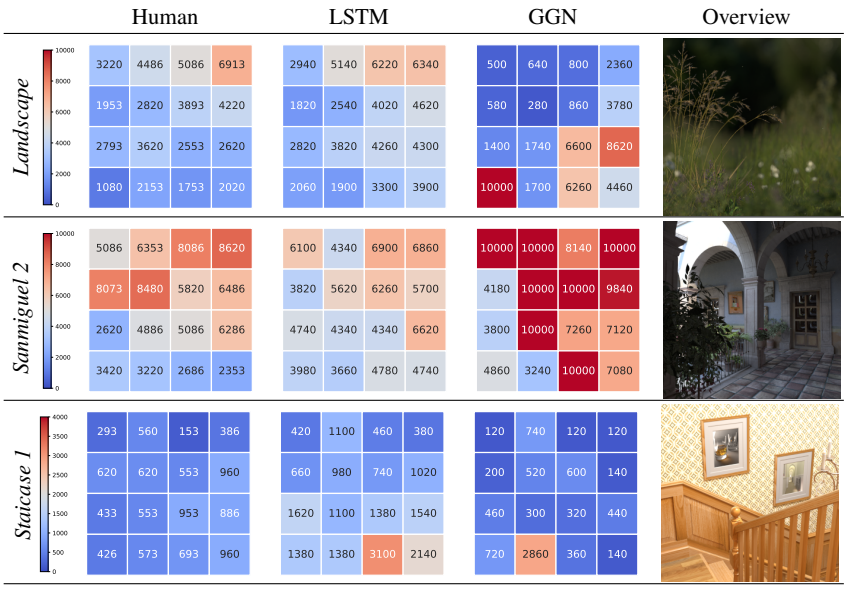


Fig. 17: Heat map comparison of LSTM and GGN model predictions on problematic scenes. For each scene the scale of values can be different in order to visually compare the heat maps of the models predictions to the human reference (expected thresholds).

that GGN has on the *Staircase 1* and *Landscape* scenes, but also to understand the visual results obtained from the *Sanmiguel 2* scene. One important thing that can be noticed by this representation is that the suggested GGN thresholds for images *Staircase 1* and *Landscape* are low in some blocks in comparison to the human reference, but also to the LSTM. Indeed, our training set was composed with only few scenes that require a low number of samples in several blocks. It would appear that the GGN model does not seem to correctly interpret this fast convergence on this kind of scene and stops too early. On the contrary, for the scene *Sanmiguel 2*, only one block has a threshold with too low samples level (block 5 with 4180 samples) and the GGN model suggests that a lot of samples are needed in some blocks. It seems that it is still enough for the perception of this image reconstructed by the predicted thresholds to lead to a perception of difference to the human reference (with more or less precision compared to the human reference image on some blocks). On the contrary, the LSTM model seems to predict too early on 8 blocks of the *Sanmiguel 2* scene, but has a higher rate than the GGN model of reconstructed images perceived as identical in the experiment (82% vs. 70% for the GGN). This must come from the fact that a single block with a still perceptible noise is more strongly discernible in the whole image, than a noise more distributed in several blocks (as for the image reconstructed by the predicted thresholds of the LSTM model).

6 Conclusion

In this paper, we propose a guided generative network (GGN) architecture that generates accurate noise detection data to guide a binary classification task. The GGN architecture is composed of 3 models: an autoencoder model that denoises the input images from the reference data, a generative model that seeks to design a noise mask from a images (whether it is still composed of noise or has been denoised), and a discriminator that distinguishes whether there is a difference between the two previously computed noise masks. This approach has been applied to the problem of noise detection in photorealistic computer graphics where Monte Carlo methods are used to compute the images. Unlike previous detection approaches that focus on finding specific noise features to improve the performance of the model, the proposed architecture automatically generates features characterising the presence of noise in an image to guide the detection model.

The GGN provides results that are just as meaningful as LSTM, if not more accurate, in predicting the number of samples required. Indeed, its decision threshold is more robust and conservative, which is a preferable behaviour in computer graphics: stopping the calculation early leads to a still noticeable noise, while stopping later allows to obtain the same (or improved) quality even if the calculation time is higher. However, it had been noticed that the GGN as well as generative models could have difficulties to generalize and propose predictions far from human thresholds on images poorly represented in the training base. Verifying the performance of the model on a larger diversity of images would be one of the first work perspectives.

Then the work carried out in this article has focused on a single Monte Carlo integrator, namely the path tracer, which is widely used in lighting simulation. Other integrators exist, allowing the treatment of complex light phenomena. However, they generate a different kind of computational noise, for which it will be interesting to study the robustness of the GGN to be able to take into account these new types of noise, either separately, or by learning the different types of noise simultaneously. A complementary aspect is the extension of the approach to the detection of noise in computer-generated image sequences, for which the disappearance of noise in each individual image may not be sufficient to ensure that temporal noise is not present when viewing the sequence. In any case, this future work will involve the generation of new image databases, which will be made available to the scientific community. Finally it would be important to consider methods to avoid such generative model to overfit over training data [22].

References

[1] Banerjee, M., Pal, N.R.: Feature selection with svd entropy: Some modification and extension. Information Sciences **264**, 118–134 (2014)
[2] Bitterli, B.: Rendering resources (2016), https://benedikt-bitterli.me/resources/

[3] Bradley, A.P.: The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern recognition **30**(7), 1145–1159 (1997)
[4] Buisine, J., Bigand, A., Synave, R., Delepoulle, S., Renaud, C.: Stopping criterion during rendering of computer-generated images based on svd-entropy. Entropy **23**(1), 75 (2021)
[5] Buisine, J., Delepoulle, S., Synave, R., Renaud, C.: Subjective human thresholds over computer generated images (Feb 2021)
[6] Buja, A., Stuetzle, W., Shen, Y.: Loss functions for binary class probability estimation and classification: Structure and applications. Working draft, November **3** (2005)
[7] Carnec, M., Le Callet, P., Barba, D.: Objective quality assessment of color images based on a generic perceptual reduced reference. Signal Processing: Image Communication **23**(4), 239–256 (2008)
[8] Constantin, J., Bigand, A., Constantin, I., Hamad, D.: Image noise detection in global illumination methods based on frvm. NeuroComputing **164**, 82–95 (2015)
[9] Dragesco, J.: High resolution astrophotography, vol. 7. Cambridge University Press (1995)
[10] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Advances in neural information processing systems **27** (2014)
[11] Haselmann, M., Gruber, D.P., Tabatabai, P.: Anomaly detection using deep learning based image completion. In: 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA). pp. 1237–1242 (2018)
[12] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
[13] Jiang, G., Kainz, B.: Deep radiance caching: Convolutional autoencoders deeper in ray tracing. Computers & Graphics **94**, 22–31 (2021)
[14] Kajiya, J.T.: The rendering equation. In: Proceedings of the 13th annual conference on Computer graphics and interactive techniques. pp. 143–150 (1986)
[15] Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
[16] Pharr, M., Jakob, W., Humphreys, G.: Physically based rendering: From theory to implementation. Morgan Kaufmann (2016)
[17] Pravin, C., Ojha, V.: 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). pp. 1094–1100 (2020)
[18] Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)
[19] Shi, H., 0003, L.W., Tang, W., Zheng, N., Hua, G.: Loss functions for person image generation. In: BMVC (2020)

[20] Takouachet, N., Delepoulle, S., Renaud, C., Zoghlami, N., Tavares, J.M.R.: Perception of noise and global illumination: Toward an automatic stopping criterion based on svm. Computers & Graphics **69**, 49–58 (2017)
[21] Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13**(4), 600–612 (2004)
[22] Webster, R., Rabin, J., Simon, L., Jurie, F.: Detecting overfitting of deep generative networks via latent recovery. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11273–11282 (2019)
[23] Whitted, T.: An improved illumination model for shaded display. In: Proceedings of the 6th annual conference on Computer graphics and interactive techniques. p. 14 (1979)
[24] Yazici, Y., Foo, C.S., Winkler, S., Yap, K.H., Chandrasekhar, V.: Empirical analysis of overfitting and mode drop in gan training. In: 2020 IEEE International Conference on Image Processing (ICIP). pp. 1651–1655. IEEE (2020)