Denoising of path traced images using Deep Neural Networks

Michele Conti 1599133
Ruixuan Xu 1909991

Paper’s approach: GAN Gaussian noise

Path tracing

Path tracing is one of the most successful technique for producing **photo realistic images using a 3D scene**.

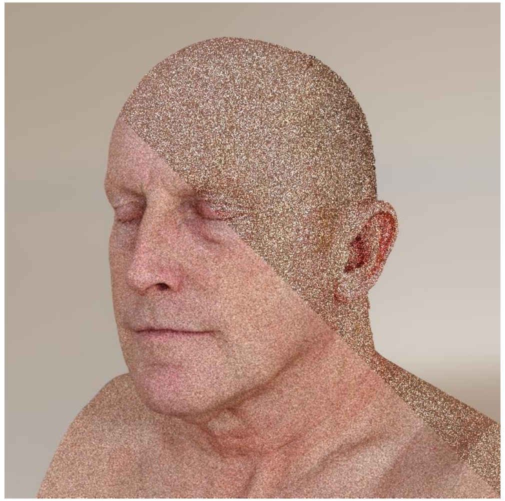
In short, the technique computes the information of **rays of light** that come from light sources in the scene and then bounce into the camera.

Monte Carlo Noise

By the nature of path tracing, it’s **impossible** to compute the path of all the possible light rays. As a matter of fact, only a handful of them are traced.

This produces some **light color variation** on images rendered with **high number of samples**: 10.000 and above number of rays per pixel.

In **low sample images** the noise is more evident, with **dark dots in parts of the image**, or places where the surface is not completely lit.



Task description

The **objective** of the model is to **remove noise** from the input image, whether it is generated using **path-tracing** techniques or it is **manually introduced**.

Why?

- It is a very **active field of research**.
- **Rendering times** can be **dramatically long**. ML approaches drastically reduce rendering times.
- Can be integrated in a **broader pipeline** (e.g., along with super resolution) for different tasks.

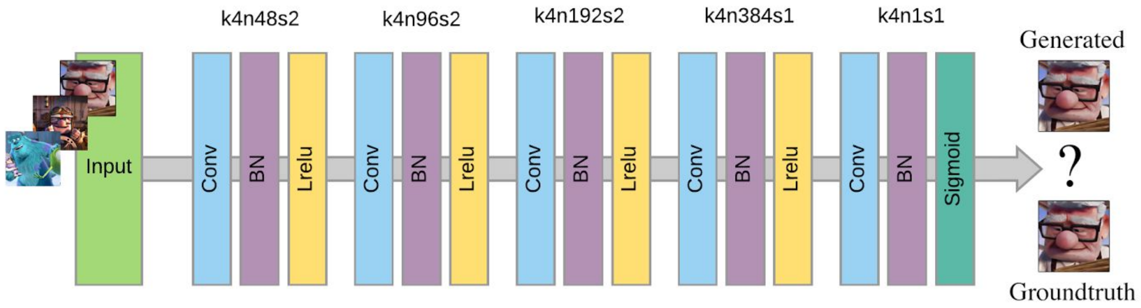
Image denoising using a GAN

The authors proposed an original approach for denoising.

The architecture used to denoise images is the **GAN**, which exploits the combination of **multiple losses** and the use of **skip connections** in the generator.

Discriminator

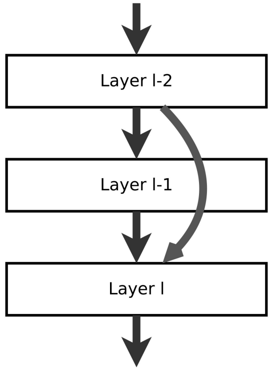
The discriminator is composed by **5 convolutional blocks**, each followed by a Batch Normalization and a Leaky-ReLU. The last layer uses instead a Sigmoid activation function.

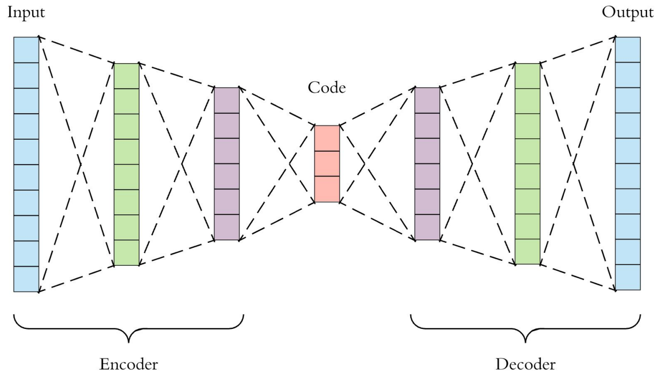


Generator

The generator is based on a **residual network** architecture, which helps in deepening the network while keeping the spatial information.

Moreover, the generator has the structure of an **autoencoder**, learning an end-to-end mapping from input noisy images to their corresponding ground-truth.

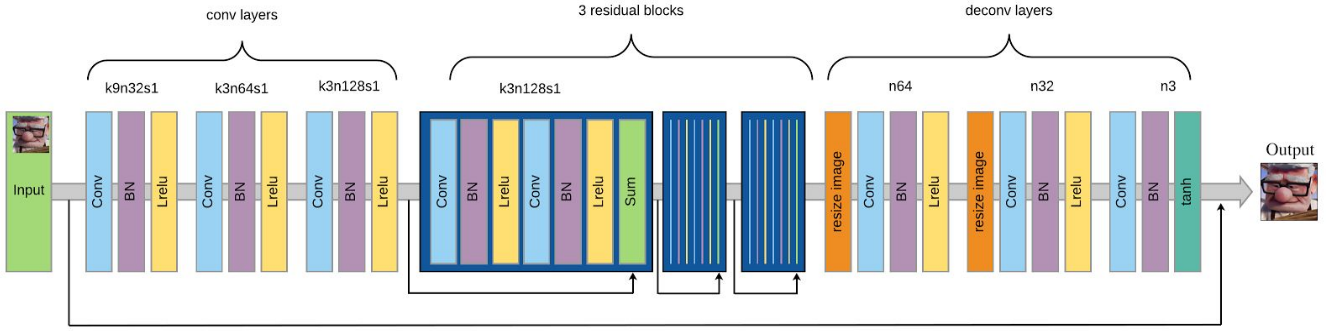




Generator

The generator is composed mainly by **3 macro-blocks**, followed by a tanh activation:

- **Convolutional** layers.
- **Residual** blocks.
- **Deconvolutional** layers.



Losses: Discriminator

It uses the standard **adversarial loss**:

It is computed by comparing the prediction of the discriminator on the **original images** with the ground truth (matrix of **ones**), and the the prediction on the **fake ones** with the opposite (matrix of **zeros**).

This results with a **matrix of probabilities**.

$$L_D = E_{x \sim p_{data}(x)}[\log(D(x))] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

Losses: Generator

A **weighted sum** of 4 losses is used to train the generator:

- **Adversarial loss**: the opposite of the discriminator loss.
- **Pixel loss**: pixel-to-pixel euclidean distance.
- **Feature loss**: pixel-to-pixel on the feature space of a pretrained model.
- **Smooth loss**: pixel-to-pixel distance with a shifted image.

$$L_G = \lambda_a L_a + \lambda_p L_p + \lambda_f L_f + \lambda_s L_s$$

Pixar Dataset

The authors manually extracted frames from **Pixar movies** to create their own dataset.

- **Training set**: 1000 256x256 images (noisy + clean).
- **Validation set**: 40 256x256 images (noisy + clean).

The noisy images are processed through the **Gaussian noise**, with varying intensity for each image. This should emulate a **light Monte Carlo noise**.

Method

Using a GAN architecture prevents the user from training until convergence, since **GANs don't have a clear convergence criteria** and losses are not really telling.

For this reason, the model has been trained until the results were **visually pleasing** and the model performed well on the validation set.

The authors chose to train their model with a batch of 7, stopping after **10.000 training iterations**.

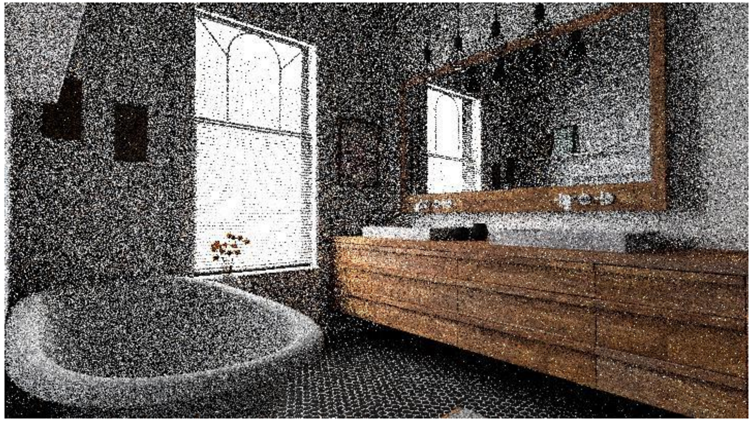
Our approach:
Autoencoder
Artificial noise

Task definition

Our main objective is to clean **path traced images** with generally low samples.

This means that the input image has a higher chance of containing **noise** that is **dissimilar to the Gaussian** one.

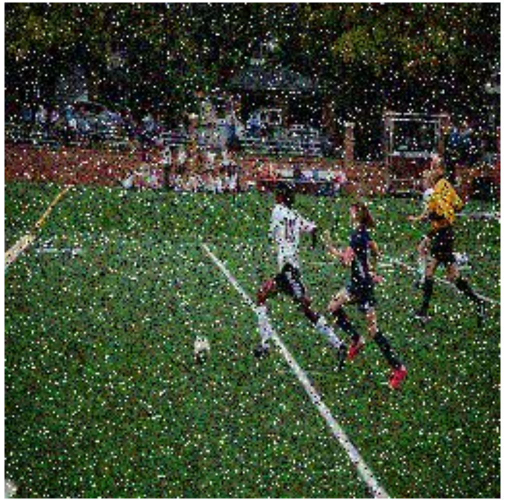
This can be a very hard task, so we set a secondary objective to **improve the results on gaussian noise**.



Artificial Noise Generation

If the Gaussian noise is enough for training a network for lightly noised path traced images, this is no longer the case for highly corrupted images.

Hence, we propose to train the model with another type of noise, **aimed to closely resemble the real path traced image**.



Salt and Pepper

One of the effect of low sample rendered images is the presence of darker dots on the image, along with possible whiter dots.

To replicate this behaviour, we introduced two artificial noises, along the gaussian one.

Pepper noise: $\hat{P}_i = k \cdot r_i \cdot P_i$

Salt noise: $\hat{P}_i = P_i + (1 - P_i) \cdot k \cdot r_i$

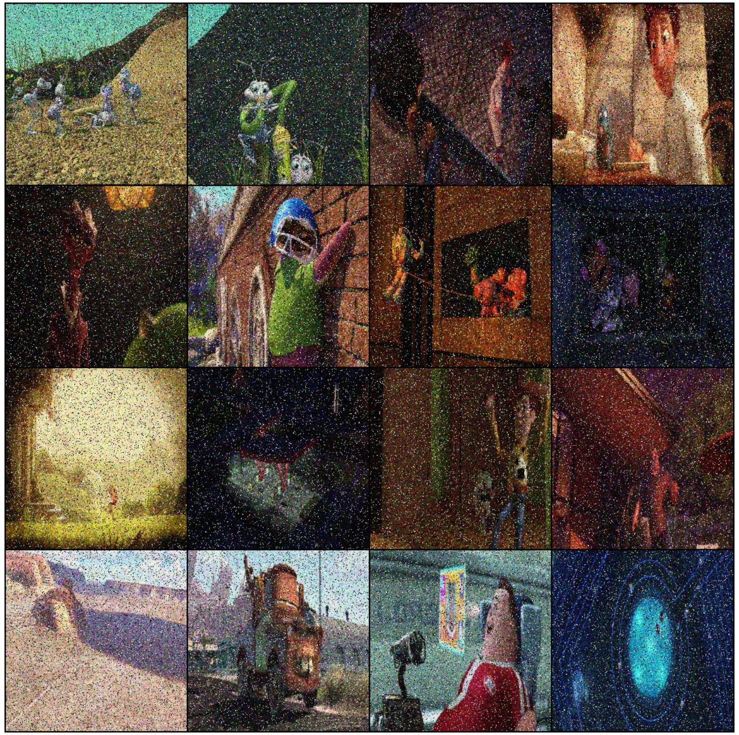
Both noises are **randomly applied**, with varying amount.

Dataset

Due to a lack of resources we kept the Pixar dataset for the training part and used two different datasets for the testing.

In particular:

- **Training**: 1000 256x256 images from **Pixar** movies.
- **Testing**: 1000 images from the COCO validation set and 40 rendered images with different degree of sampling.

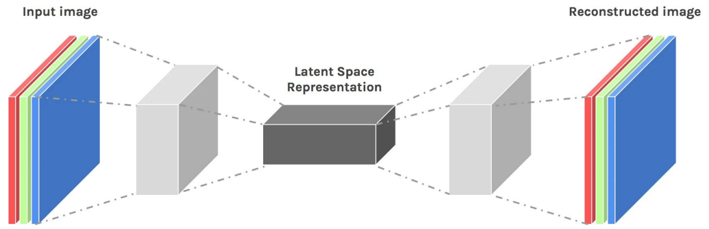


Model

One of the main drawbacks of the first model was probably the fact that the **model wasn’t deep enough**.

Still, when training GANs, there should be some sort of similarity in the structure of the two models, but **deepening both models would have lengthened the training time**.

For this reason, we switched to an **autoencoder structure**, keeping part of the residual architecture using “**residual” concatenations**.



Evaluation metrics: PSNR

Peak **S**ignal-to-**N**oise **R**atio (**PSNR**): the ratio between the maximum possible value of a signal and the power of corrupting noise that affects the fidelity of its representation.

This similarity basically represents a **scaled version of the MSE**, but it is often used in signal comparison, hence much more convenient.

$$\begin{aligned}\mathrm{PSNR} &= 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right)\\ &= 20 \cdot \log_{10}\left(\frac{MAX_I}{MSE}\right)\end{aligned}$$

Evaluation metrics: SSIM

Structural **S**imilarity **I**ndex **M**easure (**SSIM**): similarity measure based on the comparison of three features: **luminance**, **contrast**, **structure**. For image quality assessment, it is often applied **locally**, rather than globally.

$$\mathrm{SSIM}(\mathbf{x}, \mathbf{y}) = [l(\mathbf{x}, \mathbf{y})]^{\alpha} + [c(\mathbf{x}, \mathbf{y})]^{\beta} + [s(\mathbf{x}, \mathbf{y})]^{\gamma}$$

Where α, β and γ denote the relative importance of each of the metrics.

- **Luminance**: measured by averaging over all the pixel values.
- **Contrast**: measured by taking the standard deviation of all the pixel values.
- **Structure**: measured by dividing the input signal with its standard deviation.

Results

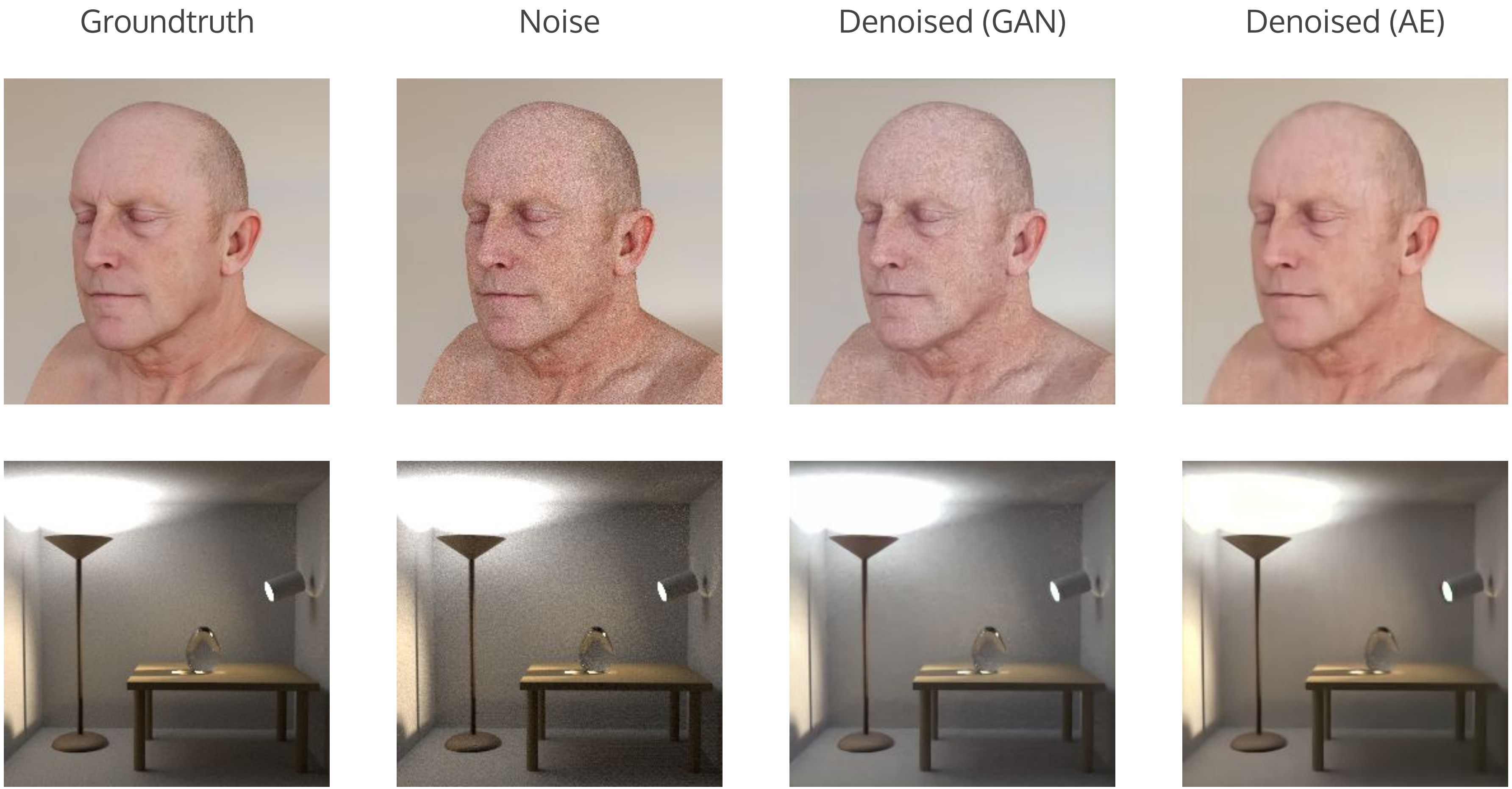
Evaluation

We compared our resulting model with the one the paper proposed, by measuring the **PSNR** and **SSIM** of both models with 3 type of noise:

- Gaussian noise.
- Our artificial noise.
- Real path traced images.

PSNR	**GAN**	**AE**
Gaussian	26.09	**26.83**
Our Noise	18.34	**25.88**
Render	22.98	**24.03**

SSIM	**GAN**	**AE**
Gaussian	**0.59**	0.58
Our Noise	0.31	**0.55**
Render	0.50	**0.51**


Groundtruth
Noise
Denoised (GAN)
Denoised (AE)

Results comparison

Evaluation metrics show that, in general, the **encoder** model works better in terms of both PSNR and SSIM, by a little amount.

Analyzing the outputs, our interpretation is that the **autoencoder** is too **aggressive** in denoising the image, while the **GAN** is more **conservative**, ignoring some minor artifact to avoid penalizations.

Moreover, the two models seem to **handle colors differently**, the autoencoder behaving a little bit better.

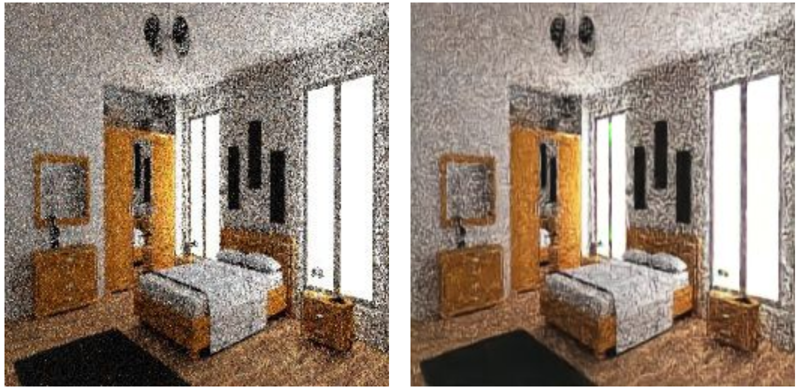
However, the **difference** between the two models is **negligible** in terms of evaluation metrics.

Limits of our model

Our model, although trained with a more detailed noise, it is far from being able to denoise some corrupted images.

With some type of scene configuration, the model fails to clean the input.

Most notable are the scenes with **higher amount of noise**, that are not seen in our dataset and hence almost completely ignored by the encoder.



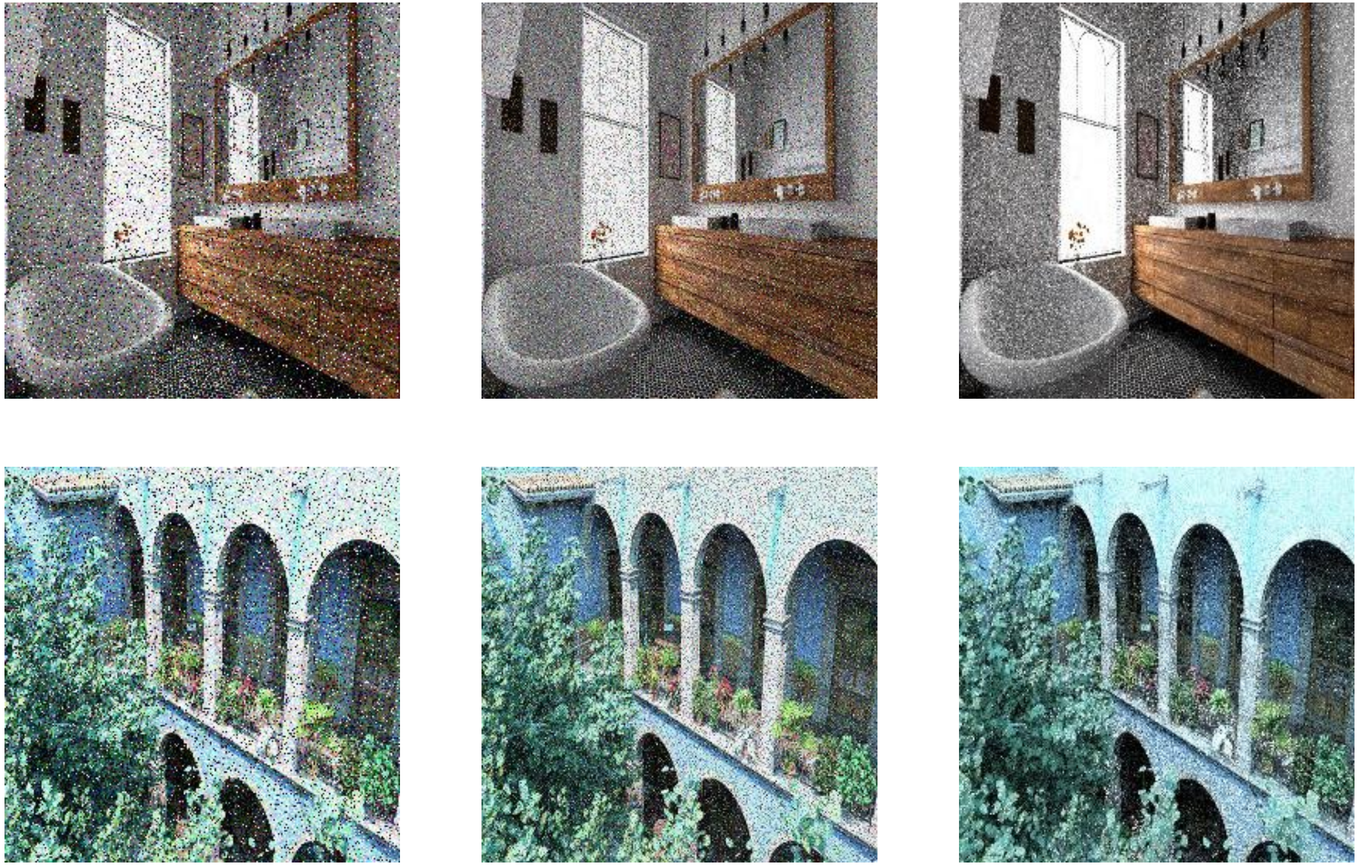
Extra: Small steps

At the end of the process we observed that applying the pepper/salt noise on single pixels was generating **very aggressive dips/spikes in luminosity**, which is **not common in rendered images**.

We tried to overcome this problem by exploiting **resizing interpolations**, so that salt & pepper **noise** was much **more subtle** along the image.

This modification further **improved performance in terms of SSIM**.

Render	GAN	AE
PSNR	22.98	23.94
SSIM	0.50	0.52


Old noise
Revised Noise
Path traced (64)

Future work

There are quite a few options one could exploit to better the performance of the models.

- Use **bigger datasets**.
- Use **real path traced images**, in addition to **auxiliary images**, such as the albedo image, and the normal image.
- Use **more sophisticated measures as loss** during the training, for example the **SSIM**.

IMAGENET

$$\mathrm{SSIM}(\mathbf{x}, \mathbf{y})$$

Conclusion

Although our **main objective is not fulfilled** (we're still not able to perfectly denoise path traced images), we were able to obtain **decent results** and even improve the ones from the original paper.

Artificial noise seems to be a **promising path**, with the help of interpolations. However, it still seems like it is almost **impossible to replicate rendering noise** algorithmically.

The task of **image denoising** is much more **data-dependant** than model/architecture-dependant.

Main references and links

[1] Image Denoising Using A Generative Adversarial Network - A. Alsaiari et al.

[2] Noise2Noise: Learning Image Restoration without Clean Data - J. Lehtinen et al.

[3] Image Quality Assessment: From Error Visibility to Structural Similarity - Z. Wang et al.

Code for **AE** model: https://github.com/mikcnt/cv-denoising-encoder
Code for **GAN** model: https://github.com/mikcnt/cv-denoising-gan