Image Denoising Using a Generative Adversarial Network

Abeer Alsaiari
University of Illinois at Chicago
1200 W Harrison St.
Chicago, IL 60607
aalsai3@uic.edu

Manu Mathew Thomas
University of Illinois at Chicago
1200 W Harrison St.
Chicago, IL 60607
mthoma52@uic.edu

Ridhi Rustagi
University of Illinois at Chicago
1200 W Harrison St.
Chicago, IL 60607
rrusta2@uic.edu

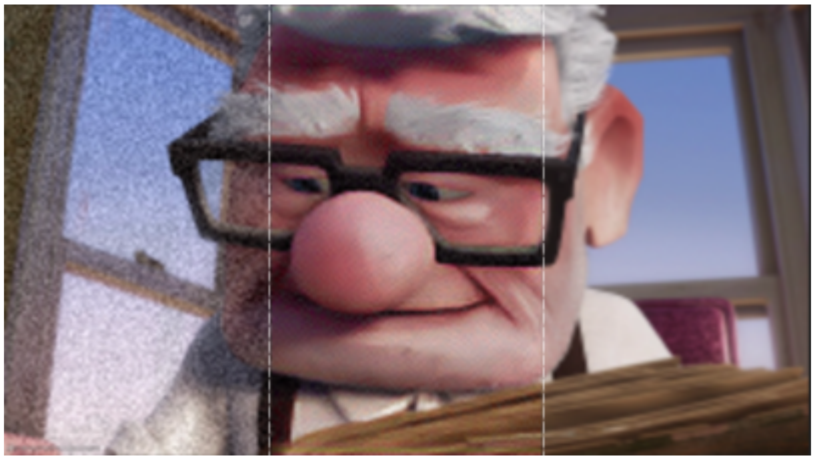


Figure 1: Gaussian noise image (left), our denoised image (middle) and ground truth photorealistic image (right).

ABSTRACT

Animation movie studios like Pixar and Dreamworks render their 3d scenes using a technique called Pathtracing which enables them to create high quality photorealistic frames. Pathtracing involves shooting 1000's of rays into a pixel randomly(Monte Carlo) which will then hit the objects in the scene and based on the reflective property of the object the rays reflect or refract or get absorbed. The colors returned by these rays are averaged to get the color of the pixel and this process is repeated for all the pixels. Due to the computational complexity it might take 8-16 hours to render a single frame.

We implemented a neural network based solution for reducing 8-16 hours to a couple of seconds using a Generative Adversarial Network. The main idea behind this proposed method is to render using small number of samples per pixel (let say 4 spp or 8 spp instead of 32K ssp) and pass the noisy image to our network, which will generate a photorealistic image with high quality.

KEYWORDS

Image denoising, Deep GAN

1 INTRODUCTION

Computer Generated Imagery became a part of daily life applications such as movies, video games and commercials. From the early stage, efforts are made to enhance the production of 3-dimensional images and many algorithms are proposed to efficiently render 3D scenes. During the 1960's and early 1970's, algorithms are proposed to render 3D scenes with enhanced realism. For example, algorithms like hidden-surface and hidden-line are proposed to resolve the visibility problem. Many other algorithms have been proposed over the years for photorealistic rendering [1]. Nowadays, animation movie companies like Pixar and Dreamworks render their 3D scenes usng a technique called Pathtracing, which enables them to create high quality photorealistic frames. Pathtracing involves shooting 1000's of rays into a pixel randomly (Monte Carlo), which will then hit the objects in the scene and, depending upon the reflective property of the object, the rays will reflect or refract or become absorbed. The colors generated by these rays are averaged to obtain the color of the pixel, and this process is repeated for all the pixels. Rendering photorealistic scenes frame by frame is very expensive and time consuming because of the need for thousands of rays per pixel and hence increased computational complexity. With the development of software and hardware, the direction moved towards more efficient rendering systems. GPUs technology, large memories and efficient software APIs brought a step further in rendering, but still nowhere close to real-time rendering. Due to computational complexity it might take 8-16 hours to render a single frame, which makes real-time rendering less feasible.

Motivated by this problem, several attempts are being made to speed up the process of obtaining high quality images. Using a few samples to render a 3D scene can be quickly evaluated, but the inaccuracy of this estimate appears as noise in the final image. This problem can be addressed using a denoising mechanism to generate a high-quality noise-free image. Image denoising methods are applicable to any noisy images either CGI, scanned images or images taken by a camera. Limitations in signal transmission equipment like cameras and scanners are the main source of noise in images. Enhancing the quality of images is important in many applications including, but not limited to, medical images and geographical pictures. In addition, a decent visual level is important for a better user experience in all applications.

The most recent promising results for many problems in image processing, including image denoising, are accomplished in particular by Convolutional Neural Networks. Artificial Neural Networks showed an outstanding performance in solving different tasks. Their performance outperforms traditional methods in numerous cases. Convolutional Neural Network (CNN) is similar in its architecture to conventional neural networks but they explicitly assume the input is an image. They have been used as an underlying architecture for image processing such as in image classification, image denoising and super resolution. Dong, et al. [2] claimed that image-denoising pipeline using example-based Super Resolution methods such as the sparse-coding-based method could be equivalent to the convolutional neural network procedure. However, sophisticated results depend on the power of the network design and training process. With respect to visual recognition tasks, the depth of the network is of central importance. The deeper the network, the better the result. Although training deep networks is very hard because of the emergence of the vanishing gradient problem, some network designs have been proposed to address this issue. Residual nets [3] are powerful networks that can handle very deep depths by heavy use of skip connections and batch normalization. They are proven to be sophisticated enough up to 152 layers of depth.

In this work, we are proposing a neural network based solution for reducing 8-16 hours to a couple of seconds using a Generative Adversarial Network. The main idea behind this proposed method is to render using a small number of samples per pixel (let say 4 spp or 8 spp instead of 32K ssp) and pass the noisy image to our network, which will generate a photorealistic image with high quality. Our proposed network is based on ResNet [3], the state of the art convolutional net. The key for our work is the defined loss function and the very deep GAN. We defined a refined perceptual loss that preserve not only color and texture but also properties of the scene like motion blur and depth of field.

The rest of the paper is organized as follows. We provide an overview of the most related work in section 2. Section 3 shows the architecture of our proposed GAN. In section 4, we discuss our experiments and achieved results. Finally, we conclude in section 5.

2 RELATED WORK

Ordinary Artificial Neural Networks have been widely used for regression problems that map continuous vectors of input to another continuous vector of output by minimizing an optimization function. In [4], a Multi-Layer Perceptron neural network is learned for image denoising formulated as a regression problem. Pairs of noisy and clean patches are used to estimate the network parameters that minimize the difference between the noisy and clean patches. Each layer applies weights to patches to be sent to the next layer.

Moving throughout a few fully connected layers, the output is compared to groundtruth value. To update the network parameters, a backpropagation is used and the mean squared error is minimized. The learned MLP network is used then to denoise images by dividing the image into overlapping patches as continuous input vectors. Each patch is denoised and the average of overlapping patches is calculated to produce the denoised image.

MLP is also used in [5] to filter out Monte Carlo noise from images. Monte Carlo rendering can produce high quality images but requires calculating many expensive rays resulting in lengthy render times. A few samples can be quickly evaluated but will produce a noisy image. They addressed this problem by applying a denoising filter to produce a pleasing high-quality image. To achieve that, they observed that the noisy image and the ideal filter parameters have a complex underlying correlation. The rendered image is a set of primary features at each pixel like screen position, color, shading, etc. A set of statistical secondary features of each pixel in relation to other pixels is achieved by processing the pixel local neighborhood. Then, a MLP network is learned with these features to output a set of filter parameters. The filter module applies the learned parameters to the noisy rendered image to filter the pixels and generate the filtered image.

Convolutional networks have been used as architecture for the image denoising process. Jain and Seung [6] proposed an unsupervised learning approach for image denoising using the convolutional network. In contrast to the typical structure of a convolutional network that outputs a single score, their network restores a denoised image from an input that is subjected to the Gaussian noise model. The network outperformed previous approaches with 4 hidden layers and 24 feature maps per hidden layer. During the unsupervised learning, the network is trained with the Berkeley segmentation dataset. This dataset contains noise free images and therefore they formulated the denoising problem as an unsupervised learning process. During the training, a noise function with different variations is integrated into the training process to synthesise noisy training samples from noise free images. The network is then trained to denoise images by minimizing the reconstruction error of noisy samples as a loss function. That is, it's trained to minimize the difference between the free noise image and the noisy reconstructed image.

Another related problem that is solved using convolutional networks is the single image super resolution. It's the processing of a low-resolution image to estimate its high-resolution counterpart. The convolutional network is proposed in [2] for single image super resolution. Similar to [6], the input image is divided into overlapping patches. Using two convolution operations, conceptually high-resolved patches are computed and then aggregated to compose a high-resolution output image. Due to the difference in resolutions, the input image is upscaled first to the desired resolution as a preprocessing step. The parameters of mapping function are learned during the training by minimizing the mean squared error between the reconstructed image and the ground truth image.

The most related GAN-based work to ours is the SRGAN, a proposed GAN for Single Image Super-Resolution [7]. Unlike previously proposed convolutional networks for image super resolution that are based on the mean squared error as an optimization function, they proposed a new loss function that resolves perceptually satisfying high-resolution image. The proposed architecture is a very deep residual net architecture, which is a GAN-based network consisting of Discriminator and Generator networks. The generator network is trained to generate an indistinguishable image from the ground truth. So, it fools the discriminator with the reconstructed image. Similarly, the discriminator is trained to discriminate the reconstructed image from the real image. The training of the network is achieved by minimizing the perceptual loss function, that is, minimizing the weighted sum of its components: content loss and adversarial loss. Instead of relying on pixel-wise error measures such as MSE-based optimization, they proposed a novel perceptual loss function consisting of content loss and adversarial loss. The goal of adding content loss is to handle the solution with respect to perceptual high-level features. The content loss is based on perceptual similarity. Using pre-trained 19-layers VGG network, feature maps are obtained for the reconstructed and reference images. The feature map is computed by encoding each image vector by layer filters. The difference between features maps of reconstructed images and reference images is computed as a Euclidean distance to define the content loss. The adversarial loss makes the discriminator and generator push the solution to the natural image space in which the generator tries to fool the discriminator with the reconstructed image and the discriminator tries to distinguish the reconstructed image from the real image. The architecture is based on ResNet [3], the state of the art network for very deep convolutional networks. ResNet is defined to ease the training of very deep networks by adding shortcut connections that feed the input to deep layers. The residual layers then retain spatial information and tune the output with reference to the input.

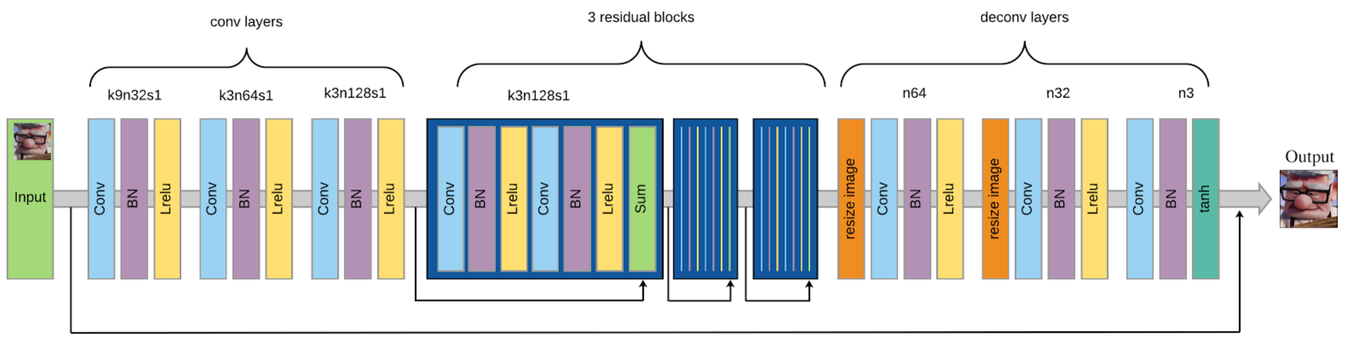


Figure 2: Generator Network

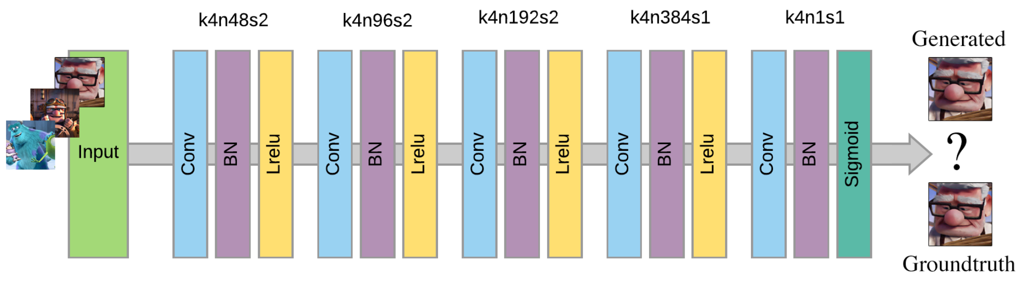


Figure 3: Discriminator Network

3 METHOD

We built a GAN network based on ResNet [3] structure for image denoising. It's aimed to make a generator network to denoise images by generating a noise free image while competing a discriminator on a ground truth reference to improve the generated image. ResNet [3] are the state of the art convolutional net for very deep CNN. The residual block concept with the use of skip connections and batch normalization can assure very promising results with good training. Due to training time limitation we only used three residual blocks. Having a larger number of residual blocks would increase the training accuracy significantly with the expense of more training time.

3.1 Generator Network

The goal of single image denoising is to generate a photorealistic image with high quality. The generator should be able to fill in the noises with the neighboring pixel colors as much as possible without loosing any detail information of the original image. So the key part lies in designing a good structure to generate denoised image.

We adopt a symmetric structure, which is similar to traditional CNN frameworks to directly learn an end-to-end mapping from input noisy image to its corresponding ground truth. A set of three convolutional layers along with Batch Normalization and Lrelu activation are stacked in the

Then there are three residual blocks each containing two convolutional layers with Batch Normalization and Lrelu activation to increase the depth of the network. We involve symmetric skip connections in this sub-network to make the network efficient in training and have better convergence performance. Those skip connections feed the input to the deep layers so each residual layer tune the output with reference to input and retain spatial information. These are followed by three sub-pixel convolutional layer each corresponding to the convolutional layers in the front of the network. Each sub-pixel convolutional layer consists of a resize image block followed by convolutional layer. The images are resized from resized from 64X64 to 128X128 and final image output is of size 256X256. We use sub-pixel convolutional layers instead of deconvolutional layers to avoid checkboard like patterns in the image. Since the sub pixel convolution is similar to deconvolution, we will refer to those layers as deconvolutional layer in this paper. The first two deconvolutional layers have Lrelu activations and the final deconvolutional layer has a sigmoid activation, which gives the final output. For all layers we use stride 1. The generator network is as follows:

*CBL(K) – CBL(K*2) – CBL(K*2*2) – CBL(K*2*2) – CBL(K*2*2) – CBL(K*2*2) – DBL(K*2) – DBL(K) – DB(3)-Tanh* where, CBL(K) is a set of K-channel convolutional layers followed by Batch Normalization and Lrelu activation, DBL(K) is a set of K-channel deconvolutional layers followed by batch normalization and Relu activation. Skip connections are added via every two skip.

3.2 Discriminator Network

The goal of denoising a noisy image is not only to make the denoised result visually appealing and quantitatively comparable to the ground truth, but also to be of photorealistic high quality. Therefore, we included a learned discriminator sub-network to classify if each input image is real or fake. We use five convolutional networks with Batch Normalization and Lrelu activation throughout discriminator network.

Once we calculate the learned feature from a set of these Conv-BN-Lrelu, a sigmoid function is stacked at the end to map the output to a probability score normalized to [0,1]. The structure of the discriminator sub-network is as follows:

*CB(K2)-CBL(K2*2)-CBL(K2*2*2)-CBL(K2*2*2*2)-C(1)-Sigmoid*

where, CB(K2) is a set of K2 channel convolutional layers followed by batch normalization and C(1) is a set of 1-channel convolutional layers.

3.3 Refined Loss Function

To ensure that the results have good visual and quantitative scores along with good discriminatory performance, we propose a new refined loss function. Specifically, we combine pixel-to-pixel Euclidean loss (pixel loss), feature loss, smooth loss and adversarial loss together with appropriate weights to form our new refined loss function.

Adversarial Loss is for the generator to produce better output to fool the discriminator. Pixel Loss is to correctly fill the noise with colors by comparing every pixel of generated image with the ground truth image (Euclidean distance). Feature Loss is to extract features accurately and is calculated the same way as Pixel loss but between the image data extracted from the Conv2 layer of VGG16 network. We add a new loss to these existing loss functions called Smooth Loss. The intuition behind it is to prevent major difference between the neighboring pixels which can cause checkboard pattern in the image. To calculate the smooth loss we slide a copy of the generated image one unit to the left and one unit down and then take an Euclidean distance between the shifted images. The new loss function is then defined as follows:

$$L = \lambda a La + \lambda p Lp + \lambda f Lf + \lambda s Ls \qquad (1)$$

where, where LA represents adversarial loss (loss from the discriminator D), LP is pixel loss (pixel-to-pixel Euclidian distance between generated image and ground truth image), Lf is Feature Loss (pixel-to-pixel Euclidian distance between generated image and ground truth image from the Conv2 layer of VGG16) and Ls is Smooth loss. λa, λp, λf and λs are pre-defined weights for adversarial loss, pixel loss, feature loss and smooth loss, respectively.

4 EXPERIMENT AND RESULTS

In this section, we present details of the experiments and we also discuss the dataset and training details.

4.1 Dataset and Training

Due to the lack of availability of large size datasets for training and evaluation of single image denoising, we synthesized a new set of training and testing samples in our experiments. We downloaded 40 Pixar movie image frames and added Gaussian noise to create the dataset. It is ensured that Gaussian noises of different standard deviation are added to generate a diverse training and test set. Each image denoised with five different amount on noise. The training set consists of a total of 1000 images and the test set has 40 images. The images with noise form the set of observed images and the corresponding clean images form the set of ground truth images. All the training and test samples are resized to 256×144.

4.2 Model details and parameters

The entire network is trained on aws p2.xlarge GPU using the tensorflow framework. We used a batch size of 7 and number of training iterations of 10k. During training, we set λa = 0.5, λp = 1.0, λf = 1.0 and λs = 0.0001. We set K = 32 and K2 = 48 for the proposed generator and discriminator. The first convolutional and deconvolutional layer of the Generator (G) is composed of kernel of size 9 with stride 1. All the other convolutional and deconvolutional layers in the Generator are composed of kernel of size 3X3 with stride 1. All convolutional and deconvolutional layers in the first three layers of the discriminator (D) are composed of kernels of size 4×4 with a stride 2 and zero-padding by 1. The last two layers in D are composed of kernels of size 4×4 with a stride 1 and zero-padding by 1.

Figure 4: Results – Test Set of Digital Images

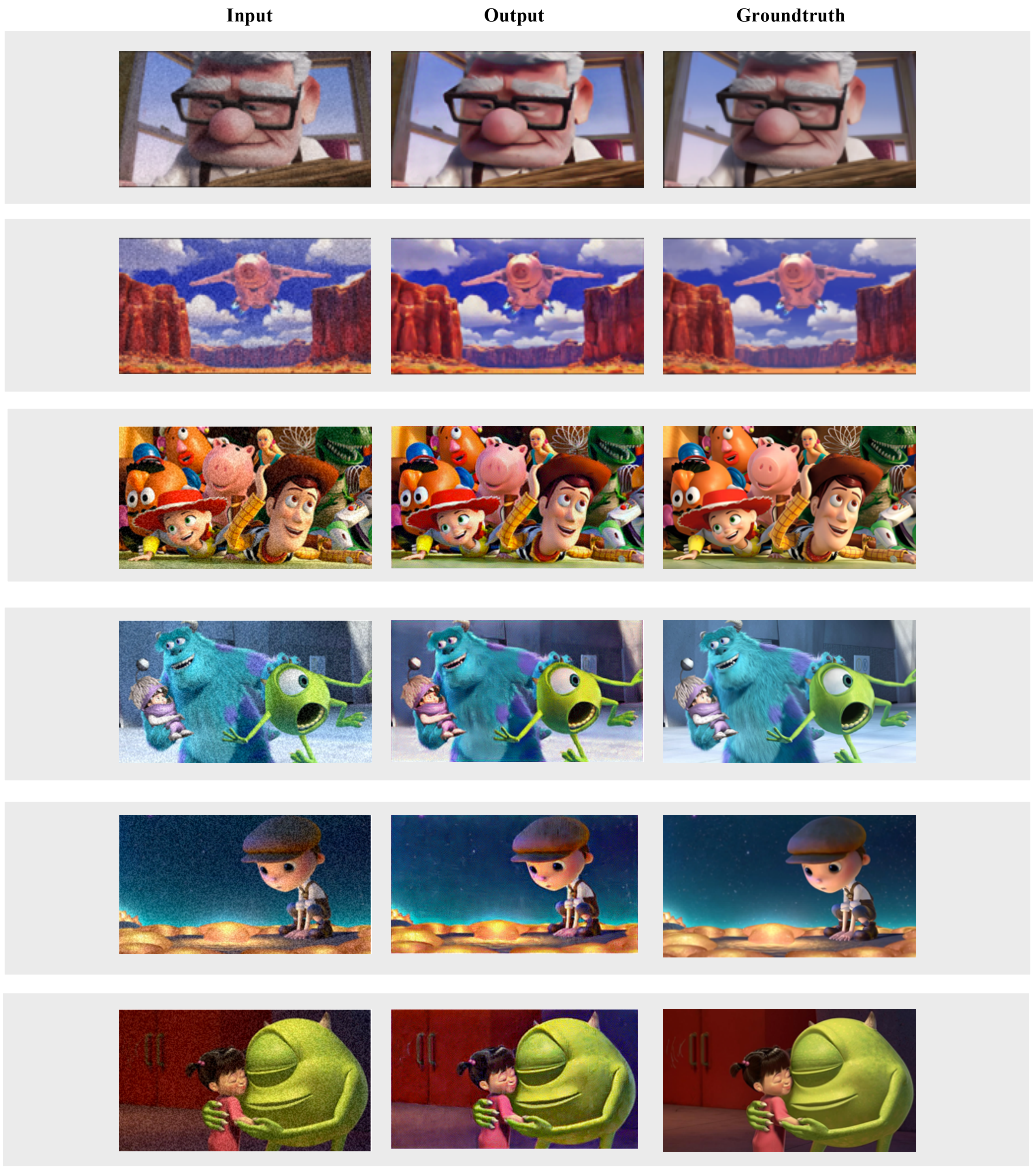


Figure 5: Results – Photos from digital camera under natural light

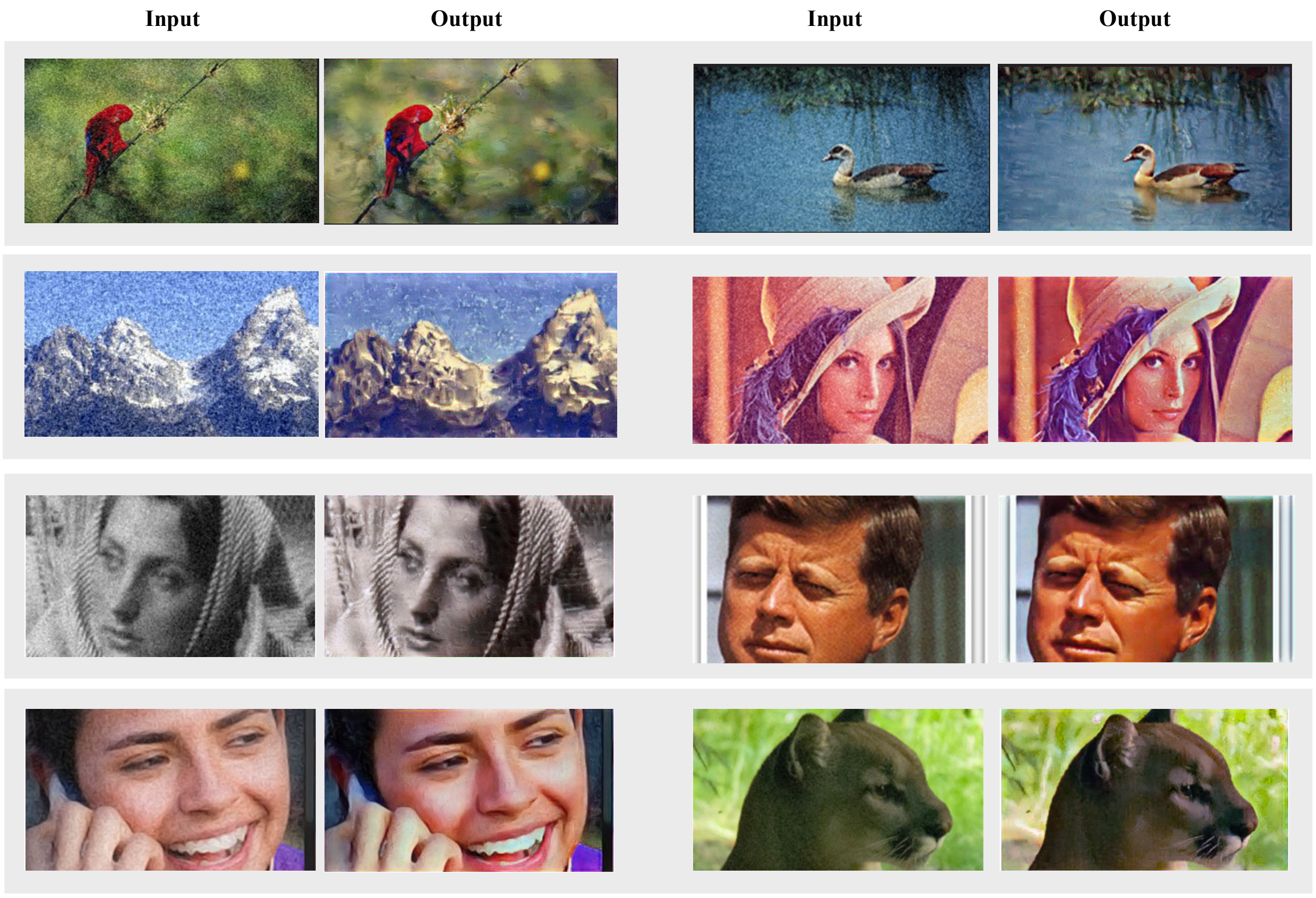


Figure 6: Result –CT Scan

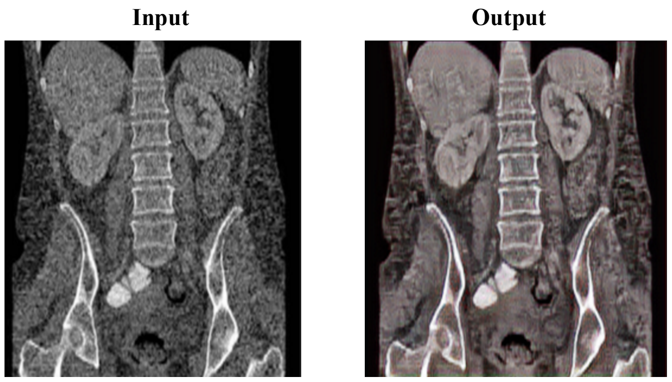
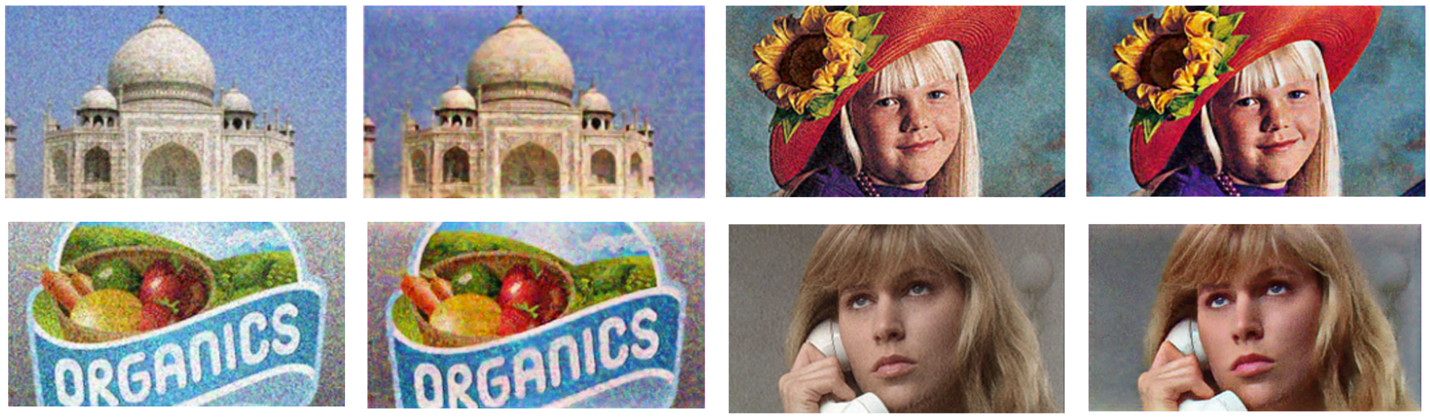


Figure 7: Results – Partially working results based on the noise unknown to the network



Colored Gaussian noise **Unknown noise**

5 CONCLUSIONS

In conclusion, we developed a novel technique to perform image denoising which takes advantage of a generative adversarial network. Our network can take a noisy image and generate a denoised version in a couple of seconds preserving edges and avoiding blurriness. In this implementation, we trained our network with Gaussian noise but it can be extended to handle other types of noise both uniform and non-uniform depending on the training dataset. Interestingly, we found that our network, trained with 40 images from a specific domain and trained for only 10K iterations was capable of denoising images outside the domain it was trained. The network gave impressive results when we gave a real noisy photo, noisy CT scan, and a noisy video.

In future, we plan to include noises produced by Monte Carlo rendering. This is will allow us to investigate whether our technique can be used for making an efficient real-time path tracer which can be used for games or medical viewer apps. As of now, we generate the denoising image based on the available pixel color information, we would like to examine if the network can be extended to fill in the noisy area based on the semantics of the neighboring pixel or by giving the network with additional information like depth map of the scene. Finally, we would also like to investigate whether our network can perform denoising with motion blur, depth of field, shadows, caustics and global illumination. Though the results are very promising for digital images, the trained network doesn't perform well against unknown noise as seen in Fig. 7. Its limited to monochromatic noise.

ACKNOWLEDGMENTS

This project is part of UIC Multimedia Systems class instructed by Prof. Angus Forbes. We would like to thank the professor for his continuous support and guidance, which aided us to explore various GAN_based applications that help us to accomplish this project successfully.

REFERENCES

[1] "3D Rendering History: Part 1. Humble Beginnings Cgsociety". *The CGSociety*. N.p., 2017. Web. 1 May 2017.

[2] Dong, C., Loy, C.C., He, K. and Tang, X., 2016. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, *38*(2), pp.295-307.

[3] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770-778).

[4] Burger, H.C., Schuler, C.J. and Harmeling, S., 2012, June. Image denoising: Can plain neural networks compete with BM3D?. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (pp. 2392-2399). IEEE.

[5] Kalantari, N.K., Bako, S. and Sen, P., 2015. A machine learning approach for filtering Monte Carlo noise. *ACM Trans. Graph.*, *34*(4), p.122.

[6] Jain, V. and Seung, S., 2009. Natural image denoising with convolutional networks. In *Advances in Neural Information Processing Systems* (pp. 769-776).

[7] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z. and Shi, W., 2016. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*.

[8] Zhang, H., Sindagi, V. and Patel, V.M., 2017. Image De-raining Using a Conditional Generative Adversarial Network. *arXiv preprint arXiv:1701.05957*.

[9] Yeh, R., Chen, C., Lim, T.Y., Hasegawa-Johnson, M. and Do, M.N., 2016. Semantic Image Inpainting with Perceptual and Contextual Losses. *arXiv preprint arXiv:1607.07539*.