International Journal of Engineering & Technology, 7 (3.24) (2018) 662-665

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

High Densely Connected Convolutional Networks for Denoising Monte Carlo Rendering

[1]Mincheol Kim, [*2]Kwangyeob Lee

[1]Department of Computer Engineering, SeokyeongUniversity, 124, Seogyeong-ro, Seongbuk-gu, Seoul, 02713, Republic of Korea
*[*2]Department of Computer Engineering, SeokyeongUniversity, 124, Seogyeong-ro, Seongbuk-gu, Seoul, 02713, Republic of Korea*
**Corresponding author Email:kylee@skuniv.ac.kr*

Abstract

Background/Objectives: Monte Carlo renderings, which are recently used in animation and visual effects, produce realistic images but noise occurs during the ray tracing process.
Methods: In this paper, the learning is performed with only RGB channel without an auxiliary buffer such as normal, albedo, and diffuse. The performance was improved by modifying the Densely Connected Convolutional Networks, which shows excellent performance. The transition layer which has the pooling layer is removed, and the last convolution layer is used to produce a denoised image because the final layer is intended for denoising rather than classification.
Findings: It is difficult to distinguish the detail from the noise without special information during the denoising process, thus the learning convergence speed is slowed down. However, in this paper, we found that it is possible to preserve detail while removing noise by using the Densely Connected Convolutional Network to preserve the high and low features. Even if the feature map is increased, batch normalization and bottleneck layers can resolves this problem and even increases the speed of learning. As a result, our method denoised better than state-of-the-art base-filter denoiser with only the RGB channel.
Improvements: It was implemented by Tensor flow with Python on CPU i5 6600, and GTX 1080 Ti. After approximately 24 hours, it showed similar performance than the filter-based algorithm.

Keywords: *Convolutional Neural Network, Ray Tracing, Monte Carlo Renderings, Monte Carlo Denoising, Densely Convolutional Network*

1. Introduction

Multi-dimensional computation of each pixel of the image is required to render realistic images. Monte Carlo(MC) rendering technology can produce high-quality images using methods to estimate and approximate the scene function through tracing light rays for these complex computations.[1] However, when the number of samples(rays) is too small, an unacceptable noise will occur and an immense computational cost are required for noise-free images, as shown Figure 1. This eventually leads to a very long rendering time. Recently, many denoising algorithms have been researched to address this problem by rendering at a low sampling rate and removing noise through post processing.

Figure 1:. The amount of noise according to the number of samples

One of these methods is denoising with the use of filters, which removes the noise by using various auxiliary buffers(normal, depth, albedo, etc.).A number of filter-based methods have been proposed, including the joint bilateral filtering method[2], the method to estimate filter error by SURE metric[3], and the asymptotic bias analysis method[4], and these methods have shown good performances as they filter with information about the scene. However, the parameter setting requires manual configuration due to the use of fixed filters, which limits the ability to find optimal parameters. Thus, the failure to obtain optimal parameters resulted in overblurring or underblurring.

Recently, a learning-based method was proposed to obtain the parameters through machine learning, which not only produces good quality results but also quickly finds a filter to obtain the denoised image, once the learning process is over. However, it uses the multi-layer perceptron rather than the convolutional network for image processing and uses a very small amount of dataset. Hence, it still had the existing problems due to the partial use of fixed filters such as cross non-local means filter.[5]

The deep learning method, which can learn the complex relationship between input and output, can certainly obtain more optimal parameters over other methods, and most of all is fast during runtime. Therefore, a method was proposed to apply this to the Convolutional Neural Network(CNN), which is recently showing high performance in image processing and is highly capable of extracting image features. After learning in two different networks of diffuse and specular, the results are combined. Instead of using raw inputs, the diffuse is divided by albedo, logarithmic transformations for specular, and the gradient of each auxiliary buffer is obtained and entered as input. As a result, although it showed a good performance, it uses the naïve

convolutional network, and overblurring occurs when attempts are made to output the denoised image as it is, and it takes about a week to get a good result.[6]

In this paper, our proposed model learns only by RGB channel for various noise applications. However, unlike the conventional RGB channel, which has a specific range of values between 0 and 255, the learning process is unstable and color artifacts or overblurring occurs because of the high dynamic range of values, which declines the learning convergence speed. To solve this problem, the image patch size was set to a suitable small size, and used the Densely Connected Convolutional Network to preserve the high and low features of the RGB image as much as possible. As a result, it was possible to obtain results with faster learning speed while preserving more detail and removing noise than the naïve Convolutional Neural Network.

2. Deep learning Architectures and Techniques

We focuses on the deep learning model and its techniques, as the area of rendering technology and auxiliary buffer(normal, depth, albedo, etc.) for preprocessing is too large to cover in this paper. The model in this paper is based on the Convolutional Neural Network(CNN) and uses batch normalization as a way to improve learning speed and accuracy. And also modified the Densely Connected Convolutional Network(DenseNet), which recently shows excellent performance in classification, for denoising and describes how the modification is constructed.

2.1. Convolutional Neural Network and Technic

Deep learning algorithms are already used in many fields and the technique is still under study. Especially in image processing fields such as classification, detection, and segmentation, deep learning can perform similar or better than humans and the model used to do such work is the Convolutional Neural Network. It consists of basic elements such as the convolution layer, the pooling layer, and the fully connected layer.

2.1.1. Convolution Layer

The convolution layer is a key component of the CNN, a technique which is already used in image processing and signal processing. It operates the element-wise sum of product by sliding the k x k size kernel to the image in a horizontal/vertical dimension. It can extract various features from the image according to the kernel value. Although the multi-layer, in which the neuron of each layer is fully-connected, is likely to cause overfitting of the image, since CNN has the characteristics of local connectivity, it can reduce overfitting by taking advantage of the information in the receptive field of the image. In addition, the parameter values are much lower than that of the fully connected layer because the kernel values are shared.

2.1.2. Pooling Layer

The pooling layer reduces the amount of computation by reducing the size of the feature map, which is increased by the convolution layer. In general, a 2 x 2 kernel slides in a horizontal/vertical dimension and selects a value by a certain rule such as max pooling which selects the maximum value, average pooling which obtains the average value, and max pooling is widely used in CNN currently.

2.1.3. Fully Connected Layer

The fully connected layer is used for classification by placing it on the last layer of CNNs in the same form as the MLP. The three-dimensional feature map, which is drawn through the convolution layer and the pooling layer, is spread out into one dimension and to obtain the sum of product, and the final predict value is obtained after applying the activation function.

2.2. Batch Normalization

Every time the deeper model is applied to learning, the gradient vanishing/exploding problems arise, and there were efforts to solve this problem in various ways. For instance, there were methods that initialized the weight or learning rate well, used ReLU function as activation function, and regularization such as dropout or weight decay to prevent weight values from being concentrated on one side. However, the learning speed would slow down or gradient vanishing/exploding would occur again since the fundamental problem was not solved. This instability occurs because of the internal covariance shift, and as the value is transferred to the layer, the distribution of each activation input is changed. This becomes more biased as the layer is transferred, resulting in the overfitting. To resolve this problem, each layer is normalized by calculating the moving average to remove this bias. Eventually, it does not fall into the local minima even if the learning rate is set to high through normalization, and it learns well without dropout because of the self- regularization effect. [7]

2.3. Deep Residual Network

However, there was no way to get any deeper than this due to the Vanishing/Exploding Gradient problem. Of course, this problem was solved by various methods, but if a certain layer is passed the learning process doesn't work well. But it's not overfitting because training error increases by degradation problem. In order to solve this problem, a shortcut connection is designed directly from the input to the output to learn the difference between the input and the output as shown in Figure 2, so that even small fluctuations can be easily detected. This is called residual learning, and deep networks can be easily optimized while performance can be improved because the depth can be increased easily. As a result, it became possible to increase the number of layers to more than 100 showing better performance than humans.[8]

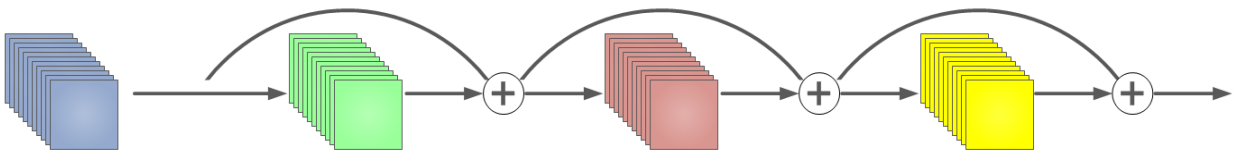

Figure 2:. TheConnectityof Deep Residual Networks

2.4. Densely Connected Convolutional Network

Although ResNet has already shown sufficient performance, a model has emerged that retains the performance while the size of the model is smaller and narrower. This is called the Densely Connected Convolutional Network(DenseNet), and DenseNet-BC(DenseNet + Bottleneck + Compression) has only 1/3 of the parameters compared to ResNet. Unlike ResNet, which adds the

block input values and operation result values, DenseNet concatenates all previous blocks, as shown in Figure 3.Other than this, bottleneck layers are used to reduce the computational complexity for a similar channel reduction. And to make the model more compact, convolution 1 x 1 and pooling operation are performed to reduce the amount of feature map. Finally, classification is performed by average pooling.[9]

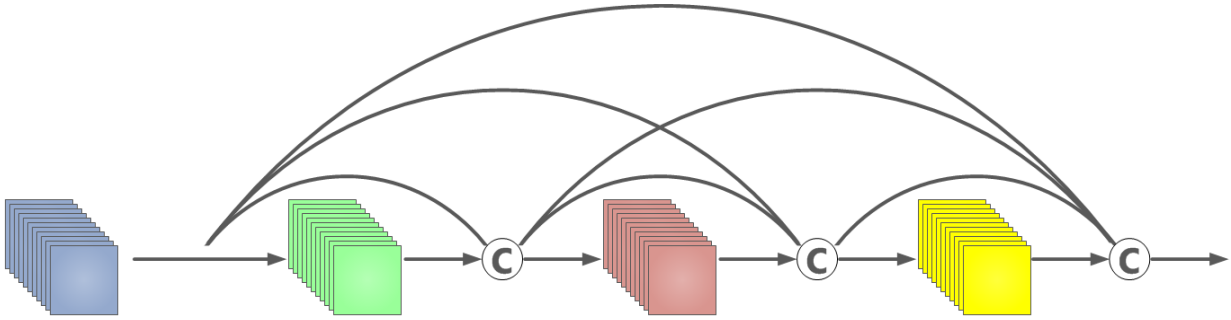

Figure 3:. TheConnectity of densely connected Convoutional Network (Dense Block)

3. Proposed Method

In this paper proposes a model that performs better than the conventional naïve CNNs by modif-ycationsDenseNet properly for denoising and faster learning convergence speed. And we also design loss function by using gradients of RGB channels. And finally we describe used parameters of our mod-el.

3.1. Densenetfor Denoising

First, the transition layer has been removed to have a bottleneck structure, while not reducing the image size and the number of feature maps. In addition, the operation of the final layer performs convolution, because the final output must be an image since the purpose is for denoising rather than classification. For increasing learning speed and reduces overfitting, we use 20bottleneck layers in a dense block: bottleneck layer consists of Batch Normalization(BN)-ReLU-Conv(1x1)-BN-ReLU-Conv(3x3) in order. In Figure 4, the important thing is that the proposed model has only one dense block for preserving as many feature map as possible, and the block operates as shown in Figure 3. The first layers(Conv, BN, ReLU) is extract simple features about noisy image by convolutioning with 7 x 7 kernel. The last layers(BN, ReLU, Conv) outputs the result with RGB channel through 3 × 3 kernel operation.

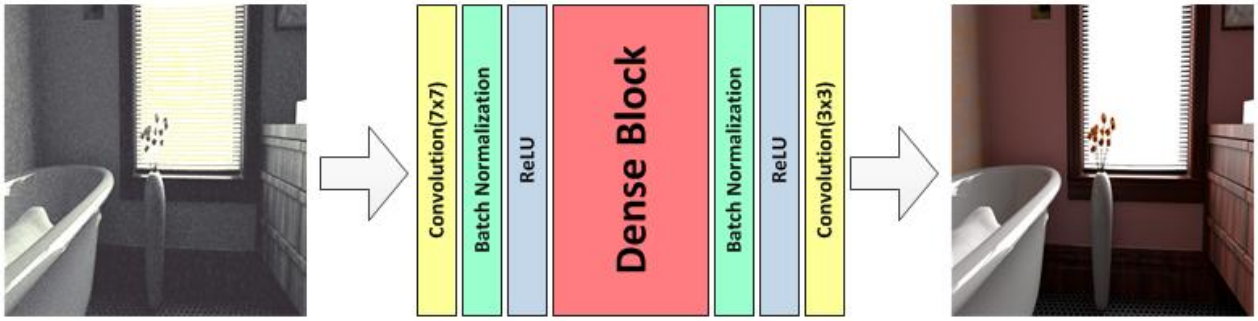

Figure 4:. The proposed model for denoising

3.2 Training

It is also known that using L_1 loss perfo-rms better than L_2 by reducing splotchy artifacts[10]

$$\mathcal{L}_s = \frac{1}{n}\sum_i^n |D_i - R_i| \quad (1)$$

In Equation (1) Where D_i and R_i are the ith pixel of denoised image and reference image respectively. But if there is only L_1 loss. There is no way for model to learn where the edge of images is. Hence, we input additional 6 channels about gradient of RGB channels(x, y). And then another loss is calculated by subtractin gradients of denoised images and reference images. In Equation (2), g(·) is computed difference on x, y-axis.

$$\mathcal{L}_g = \frac{1}{n}\sum_i^n \left| g_{x,y}(D_i) - g_{x,y}(R_i) \right| \quad (2)$$

We use these two losses for computing final losses by weighting. In Equation (3), we picked $w_{s/g}$= 0.9/0.1. Adam Optimizer was used as the optimization function. Learning rate starts from 0.01, decay step 10000, and decay rate 0.9. And kernel initializer is used He Initializer.[11]

3. Experiment

3.1. Dataset

To avoid overfitting during the learning process of a large deep neural network, a large enough representative dataset is required to address the general denoising problem. In this paper, the dataset was provided by Pixar research and used about 1,300 rendering scenes with the size of 1,280 x 720. Each scene is a rendering of 8 different scenes(bathroom, staircase, etc.) into different types of images by changing various camera angles, textures, etc. For instance, same scene of the bathroom was created into different images, as shown in Figure 5. In this paper, these images are cropped into 20 slides in a 28 x 28 format, and by excluding images with a unique value less than 100, images with no special edge were excluded as much as possible.

Figure 5:. The various scenes of bathroom

3.2. Environment

The learning was conducted with CPU i5 6600, Memory 16MB, GPU 1080 Ti, and progressed with python 3.5 tensorflow version 1.6. The learning time was about 24 hours, and GPU memory of 8,558MB was used. It was compared with the learning-based filtering experiment. All of the inputs were used as 128spp rendered images.

3.3. Result

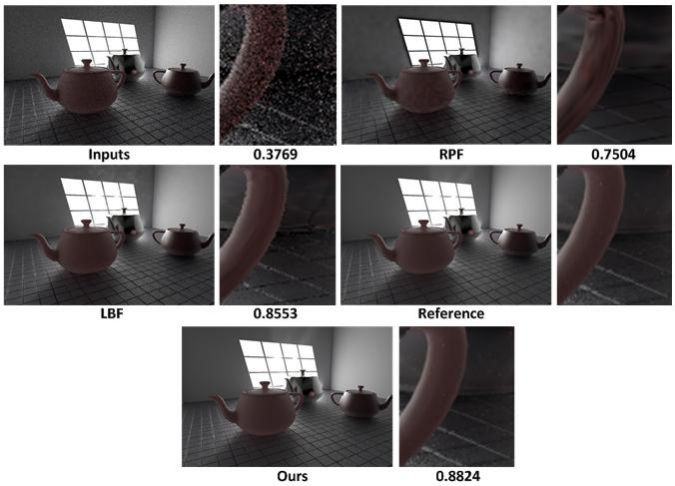

Figure 6:. The performances of denoisers

As shown Figure 6. The Result of ours performed better than RPF, LBF without auxiliary buffers unlike traditional methods. In addition, denoising time is spent only few seconds.

4. Conclusion

In this paper, among various algorithms that remove the noise of Monte Carlo renderings, a solution that applies deep learning was proposed to solve the issue. In particular, unlike the filter-based and learning-based methods, sufficient performance has been achieved with only RGB channels without special preprocessing and the use of information such as auxiliary buffers(diffuse, specular, depth).In addition, the slow learning convergence speed problem of the existing naïve CNNs and the color artifacts or overfitting problems caused by noise were solved by appropriately optimizing DenseNet, which preserves features while using layers for a long period, for denoising. However, since overblurring still exists, better performance improvements are expected if learning-based filtering is performed by preprocessing using DenseNet and auxiliary buffers.

Acknowledgment

This research was supported by the MOTIE(Ministry of Trade, Industry & Energy) (10080568) and KSRC(Korea Semiconductor Research Consortium) support program for the development of the future semiconductor device and supported by Seokyeong University in 2017.

References

[1] Terzopoulos, D., Platt, J., Barr, A., & Fleischer, K. (1987). Elastically deformable models. ACM SiggraphComputer Graphics, 21(4), 205-214.

[2] Sen, P., &Darabi, S. (2012). On filtering the noise from the random parameters in Monte Carlo rendering. ACM Trans. Graph., 31(3), 18-1.

[3] Li, T. M., Wu, Y. T., & Chuang, Y. Y. (2012). SURE-based optimization for adaptive sampling and reconstruction. ACM Transactions on Graphics (TOG), 31(6), 194.

[4] Moon, B., Carr, N., & Yoon, S. E. (2014). Adaptive rendering based on weighted local regression. ACM Transactions on Graphics (TOG), 33(5), 170.

[5] Kalantari, N. K., Bako, S., & Sen, P. (2015). A machine learning approach for filtering Monte Carlo noise. ACM Trans. Graph., 34(4), 122-1.

[6] Bako, S., Vogels, T., McWilliams, B., Meyer, M., Novák, J., Harvill, A., ...&Rousselle, F. (2017). Kernel-predicting convolutional networks for denoising Monte Carlo renderings. ACM Transactions on Graphics (TOG), 36(4), 97.

[7] Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann.

[8] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778)

[9] Huang, G., Liu, Z., Weinberger, K. Q., & van der Maaten, L. (2017, July). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (Vol. 1, No. 2, p. 3).

[10] Zhao, H., Gallo, O., Frosio, I., &Kautz, J. (2017). Loss functions for image restoration with neural networks. IEEE Transactions on Computational Imaging, 3(1), 47-57.

[11] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE international conference on computer vision (pp. 1026-1034).