Image Denoising Using Autoencoders

Doğa Yılmaz
Özyeğin University
doga.yilmaz.11481@ozu.edu.tr

Assist. Prof. Dr. M. Furkan Kıraç
Özyeğin University
furkan.kirac@ozyegin.edu.tr

Abstract

In today's world digital photography is a crucial part of our lives. Most of those images are taken with mobile devices such as smartphones. Due to their compact design there are limited space for imaging equipment, thus high-quality components used in DSLR cameras are not used in compact devices. Those small sized components used in compact devices inevitably causes noisier images. Another point is while storing images usually compression algorithms are used in order to save from space. Those methods are also causing an increase in the amount of noise. To reduce the effects of the noise visibly, denoising algorithms are used. This project aims to study and implement several deep neural network based denoising methods in order to understand their working principles and comparing the performance of autoencoder based deep models with the state-of-the-art methods. At this project smartphone image denoising dataset (SIDD) [1] is used to satisfy the data needs. Within the scope of this project, 3 different techniques are used while implementing deep autoencoder models and those models are trained using smartphone image denoising dataset. Autoencoder architecture is mainly used in this project because of its success in the denoising task. This success of autoencoder architecture is due to its ability of setting the target values to be equal to the inputs. [10] By using this property of autoencoders when clean image x is given as input and noisy image y is given as output, noise n should be reduced due to the bottleneck created in the latent space between encoder and decoder layers. That can be also shown as if $x = y$, then $x = y + n$ if and only if $n = 0$. Moreover, within the scope of this project benchmarking is done using SSIM and PSNR metrics of the implemented autoencoder based models. According to the benchmark results, it has been seen that deeper models are performing better at denoising task. Finally, all of the results are compared with the state-of-the-art methods.

1. Introduction

With the advancing technology, almost every mobile device we have contains some sort of photo or video capturing capabilities. However, most of them are designed to be compact. Thus, they have very little room inside for the camera sensor. As a result, most of them use small sized sensors which causes an increased noise in the taken image compared to the larger sensor sized DSLR counterparts.

Today image denoising is an active research area which aims to obtain clean image x from noisy version y that can be shown as $x = y + n$ where n is the noise. This project aims develop deep neural network models which generates an output which converges to x when a noisy image y is supplied. A more detailed description of the image denoising problem will be provided in the problem statement section. As deep neural networks require big amount of data to work, a high-quality image denoising dataset is required. Due to its recency and quality, smartphone image denoising dataset (SIDD) [1] will be used to satisfy the data needs.

This project offers numerous methods to perform the denoising operation. All of the proposed methods are convolutional neural network (CNN) based since this method have shown its superior performance in image processing and computer vision tasks. [8] Also, in some of the proposed deep neural network models, residual learning is used in order to improve the denoising performance. These methods are detailly explained in the proposed solution section with their architecture and working principles.

To implement and train deep neural network models efficiently, a machine learning framework is needed. PyTorch is used in this project due to its performance and ease of use. Also, as programming language python is used because of its compatibility with PyTorch and ease of use. As data visualization tool TensorBoard is used because it provides live tracking of the desired data and also it is supported by PyTorch.

Within the scope of this project, SIDD medium

dataset has been divided as training and validation sets, data loaders for each set is implemented in order to use the data with PyTorch. After that, dataloaders have been optimized in order to avoid any bottlenecks while loading images every epoch. Following that, various deep autoencoder models are implemented using mainly 3 different techniques which will be explained in the neural network architectures section. At the table 1, a performance preview of the proposed models can be found.

Table 1: Performance preview of the models.

Model Name	Average PSNR
OursV0s	33.07
OursV1s	29.25
OursV2s	38.37

After that trainings of those models are done using GPU acceleration. Finally, according to results and benchmarks models are compared with the state-of-the-art methods.

The remaining content is organized as follows. A brief review of necessary knowledge is provided in background section. After that, image denoising task is detailly explained in problem statement section. Following, details about SIDD dataset and architecture of the proposed deep neural networks are provided in the solution approach section. Training results, testing results and comparisons with the state-of-the-art methods are provided in the results and discussion section. The last section focuses on conclusion of the project.

2. Background

This section will provide a review of necessary technical knowledge which is required in order to understand the contents deeply. Also, will include the brief explanations of tools and technologies used.

2.1. Neural Networks

Neural Networks consist of many densely connected elements called neurons, which are basic computational units. When given an input to the connected network of neurons, the input is modified by a weight and summed. By using optimization algorithms like backpropagation these weights are updated in order to make the network output as close as possible to the desired output. [5]

2.2. Convolutional Neural Networks

Convolutional Neural Networks (CNNs), are a class of biologically inspired neural networks which minimizes the loss with respect to the given input and output pairs by passing the input through a series of convolutional filters and simple nonlinearities. They have shown remarkable results in a wide variety of machine learning problems such as image denoising which is the main topic of this project. [7]

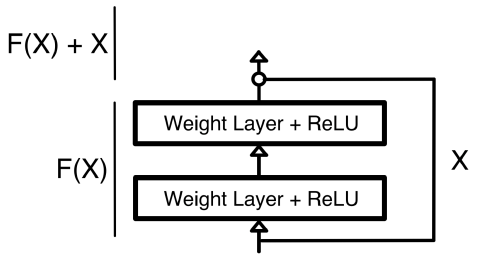

Figure 1: Sample network block with a skip connection where ∘ denotes element wise sum of feature maps..

2.3. Autoencoders

An autoencoder neural network is an unsupervised learning algorithm that applies backpropagation, setting the target values to be equal to the inputs. [10]

2.4. Residual Learning

Residual learning is a technique used in deep learning to reduce the degradation problem in deep networks. It aims to pass information directly to the deeper layers of the model by adding skip connections. An example neural network block which uses skip connection can be seen in the figure 1.

2.5. Peak Signal to Noise Ratio

Peak signal to noise ratio is an image quality metric which is widely used for comparing image enhancement algorithms in the literature. The higher peak signal to noise ratio value gets, image contain less noise.

2.6. Structural Similarity

Structural similarity is a metric that used to measure how structurally images are alike. SSIM is widely used in image processing to measure if the image is structurally corrupted after the processing it. The more structural similarity value closes to 1 the more similar the input images are.

2.7. RGB Image

RGB Images consists of 3 channels which are red, green, and blue. By combining these channels, color images are produced.

2.8. Python

Python is a high-level, general-purpose programming language. There are wide variety of libraries avalible for python including PyTorch. Due to ist flexibility and library support it is used in the implementation of this project.

2.9. PyTorch

PyTorch is an open source machine learning library. In this project it is used for implementing and training deep neural network models.

2.10. CUDA

CUDA is a parallel computation platform developed by Nvidia. It is used in this project to accelerate calculations in the training process.

2.11. TensorBoard

TensorBoard is a visualization tool for machine learning experimentation. In this project it is used to log and visualize the data in order to better understand the behavior of the deep neural network models.

3. Problem Statement

Image denoising is a well-known problem in low-level vision, which has been widely studied in the literature. Although there exist many studies about image denoising, it still remains as an active research topic. There are different categories of image denoising task such as RGB image denoising and grayscale image denoising. This project focuses on RGB image denoising. The main scope is to understand how current RGB image denoising techniques work, how to implement them and how to converge to the performance of state-of-the-art methods.

The main research question is how to obtain clean image x from noisy version y that can be shown as $x = y + n$ where n is the noise. In the literature there are some models which are designed to learn the noise n. By using this approach, they can obtain clean image x by subtracting noise n from noisy image y. However, all of the proposed architectures in this project aims to directly learn the function $f(y) = x - n$.

Measuring the performance of the deep neural network models just by looking at the output image quality is a subjective task. Thus, structural similarity metric (SSIM) and peak signal to noise ratio (PSNR) are used to measure the quality of the algorithms.

3.1. Assumptions

In order for designed deep neural network models to work as expected, the input image should contain similar noise with our training set images which are taken with several cameras under several different light conditions thus contains several different noise levels. [1]

3.2. Constraints

Computational power and GPU memory size plays an important factor while training deep neural network models. Computational power should be sufficent enough in order to complete the training sessions of each model within a feasible time. Also models and images used in a training session has to fit in the GPU memory.

To satisfy the required computational power and GPU memory, two Nvidia GTX 1080 with 8GB of video memory are used. While tuning the hyperparameters, available resources are taken into account in order to finish the training sessions.

4. Solution Approach

4.1. Design Decisions

Design decisions section is divided into two subsections to separate details of the dataset from details of the neural network architectures. In the first section the dataset is explained in detail. Following, in the second section neural network architectures and methods used are introduced.

4.1.1 Part 1: Dataset

Smartphone image denoising dataset is preferred as the source of training data due to its high quality. SIDD dataset contains 30,000 images from 10 scenes under different lighting conditions. Five representative smartphone cameras are used to capture the noisy images. Their ground-truth images are taken with high-quality DSLR cameras. Each image of the dataset has size 5312x2988. Dataset comes in 3 different sizes small medium and full size. Small dataset contains 160 image pairs, medium dataset contains 320 image pairs and full-sized dataset contains 12.000 image pairs. At the table 2, a summary of the features of SIDD dataset can be found.

SIDD also provides the ISO level, shutter speed and illuminant temperature for each photo which can be useful for denoising process. Unlike some other datasets used in the literature, noise in the noisy images

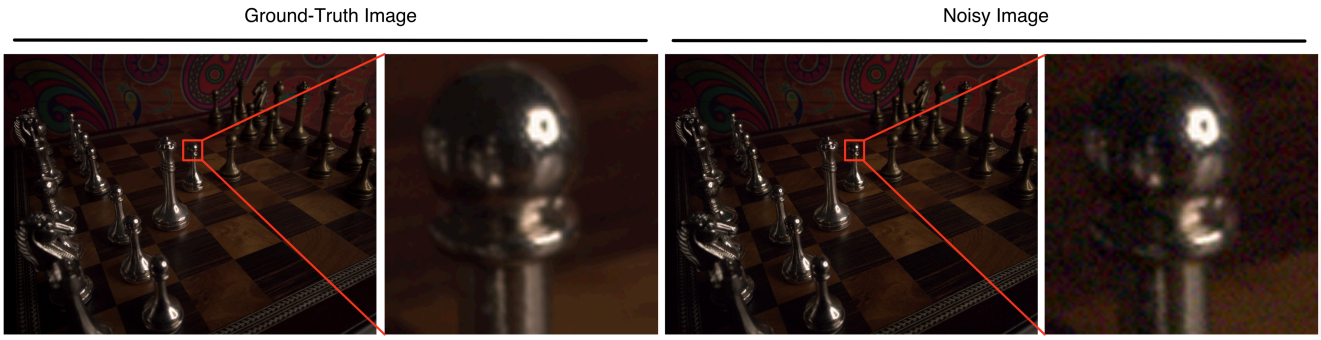

Figure 2: Example cropped ground-truth and noisy image pair.

Table 2: Features of SIDD.

Dataset Name	Image Dimensions	Image Pair Count
SIDD Small	5312x2988	160
SIDD Medium	5312x2988	320
SIDD Full-Sized	5312x2988	12000

of SIDD is the natural noise coming from the camera sensor and its lens. By training deep neural network models with natural noise, a higher performance can be expected compared with models trained with gaussian noise added images. [1]

Size of very high-quality images provided by SIDD brings some problems with it. If they are given to the network directly as an input, the number of trainable parameters exceed the limit of computational power we have today. To resolve this situation, before using images from the dataset each one is cropped in order to reduce the size of the input image. Cropping process can be seen in the figure 2. This process effectively reduces the number of trainable parameters in the neural networks.

At this point a PyTorch dataset class is needed to read the images from file and process them in the desired way. While implementing dataset class, images are saved inside a tensor object in order to reach them quickly and without causing any bottleneck.

To validate the results of implemented deep neural network models, a validation set is needed. The validation set is created from the original dataset by randomly selecting 10% of the original data.

4.1.2 Part 2: Neural Network Architectures

At this section, proposed network architectures will be introduced and explained with the detailed information of each deep learning technique used. Performace results of these models can be found in the results and discussion section. But first, naming conventions will be introduced for the sake of clarity. A sample model name can be found below. All of the models are named with same manner.

Autoencoder version 2: OursV2

OursV0

This model uses autoencoder architecture which consist of an encoder and a decoder module. As the loss function mean square error is used. This architecture is suitable for denoising task due to its ability of setting the target values to be equal to the inputs. [10] By using this property of autoencoders when clean image x is given as input and noisy image y is given as output, noise n should be reduced due to the bottleneck created in the latent space between encoder and decoder layers. That can be also shown as if $x = y$, then $x = y + n$ if and only if $n = 0$. This prosperity of autoencoders is one of the main reasons why autoencoder architecture is widely used for solving image restoration, reconstruction and denoising problems. Within the scope of this project this model is used as the base for all other models. Detailed information about the OursV0 can be found in the figure 3.

OursV0 (with skip connections)

Denoising is a highly nonlinear problem which makes learning the task challenging for the deep neural networks. To handle the complexity of this task, more layers can be added. However, as network depth is increased, accuracy gets saturated. This situation is widely seen in the literature and it is called the degradation problem. [6] In the literature, to resolve this problem, residual connections are used.

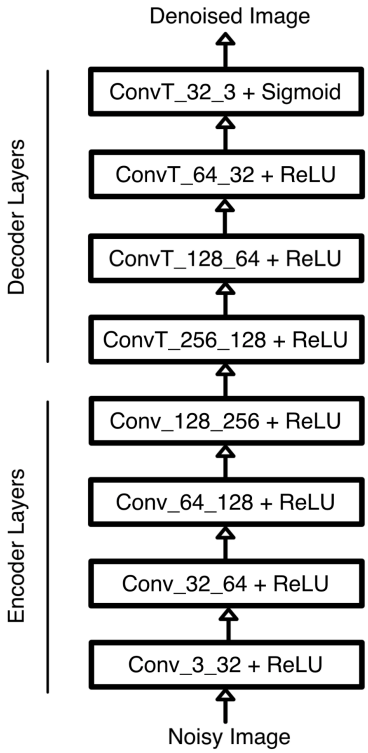

Figure 3: Visualization of OursV0 architecture.

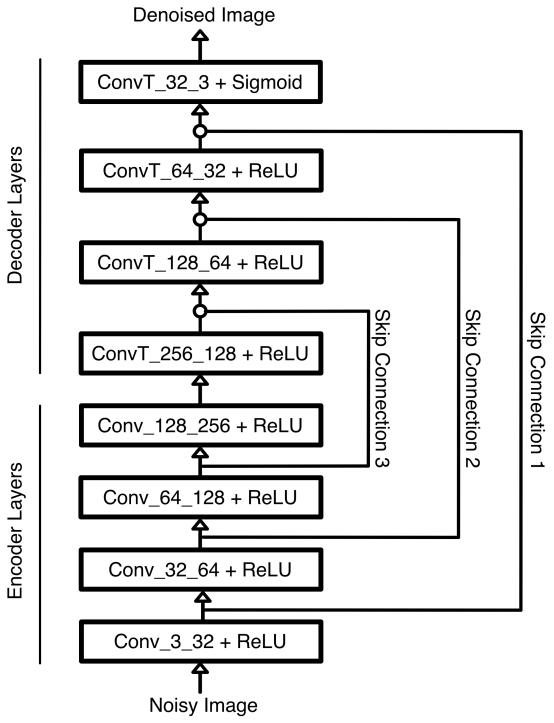

Figure 4: Visualization of OursV0 (with skip connections) architecture where ∘ denotes element wise sum of feature maps.

This model inherits OursV0 and adds residual learning on top of that in order to solve the degradation problem. As it can be seen in the figure 4 skip connections are added symmetrically.

OursV1

Number of feature map sizes considerably effects the computational power needs of the deep convolutional neural network models. When the number of feature maps increase the computational power required is also increases for both training session and making predictions. To resolve this issue number of feature maps can be reduced by using various operations such as max-pooling, interpolation and down sampleing. However, there is a tradeoff while using this approach. Omitting feature maps can cause loss of important information which is used when decoding the image. Thus, when some of the feature maps are omitted accuracy of the model slightly decreases. [8]

This model uses max pooling in encoder layers to reduce the size of the feature maps. In order to obtain an output of the same size as the input, max unpooling is used to up-sample the feature maps in symmetric decoder layers. Similarly, all models which has a version name starting with OursV1 aims to reduce the number of feature maps without decreasing accuracy to much. As the loss function mean square error is used. Since the logic of the OursV1s are very similar to each other, they do not require further explanation.

OursV2

This deep architecture consists of convolutional and transposed convolutional layers which has rectified linear unit added after each one. Unlike the OursV1s, this architecture does not contain any max pooling or max unpooling layer. As stated before, usually pooling and unpooling operations discards useful image details that are essential for these tasks. Since maximizing accuracy is the goal, every possible detail information of the image are preserved by avoiding the usage of pooling and unpooling operations. [8]

The OursV2 architecture consists of stacked blocks of convolutional layers and their mirrored transposed

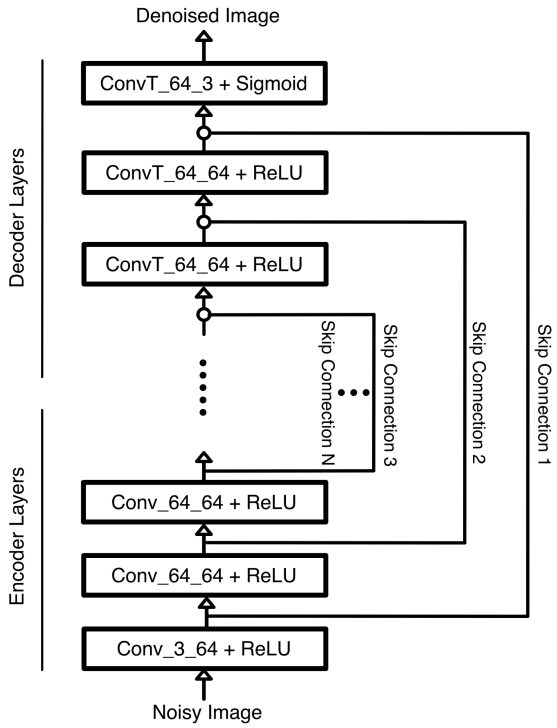

Figure 5: Visualization of OursV2 architecture where ∘ denotes element wise sum of feature maps.

convolutional layers. Also, to resolve the previously stated degradation problem, skip connections from convolutional layers to their corresponding mirrored transposed convolutional layers are added. The passed convolutional feature maps are summed to the deconvolutional feature maps elementwise and passed to the next layer. Also as the loss function mean square error is used. Visualization of the stated model can be found in the figure 5. This architecture is used with depth of 16, 20 and 30 layers. Performance results of all 3 versions of this model will be discussed at results and discussion section.

4.2. Usage of Tools and Techniques

All tools and techniques elaborated in the section two are used and helped to solve the tasks for this project. At the beginning of this project one of the main decisions was whether to use TensorFlow or PyTorch as the marine learning framework. PyTorch is selected due to its easier to understand nature. Additionally, Python programming language has been very helpful in the implementation due to its suitability of fast prototyping and development.

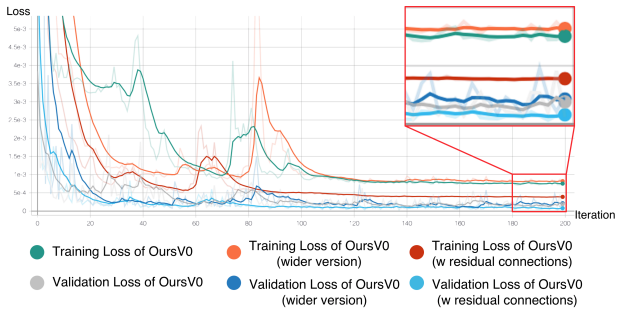

Figure 6: Training and validation loss graph of OursV0 models.

4.3. Knowledge and Skill Set

Various courses from Özyeğin University has been helpful with this project. Main and the most important courses are as follows:

- CS 423 - Computer Vision
- CS 434 - Advanced Object-Oriented Programming
- CS 321 - Programming Languages
- CS 222 - Programming Studio (Clean coding)

Beside the courses listed above all of the Computer Science courses taken from Özyeğin University were helpful.

4.4. Engineering Standards

- PEP 8 - Style Guide for Python Code

5. Results and Discussion

This section, image denoising performance of the implemented deep neural network architectures will be discussed and compared with each other using sample images which are taken from validation set as well as training loss, validation loss, PSNR and SSIM metrics.

Decreasing training and validation loss throughout the training session is one of the indications that the model is learning the task. Although it is not the only and the best method, training and validation loss values can be a good starting point when comparing the models with each other. At figure 6 the loss graph of OursV0 models is given. As it can be seen from the graph, OursV0 with residual connections has a lower training and validation loss values when compared with the ones without residual connections. By just looking at these values it can be stated that adding residual connections improves the accuracy of the model.

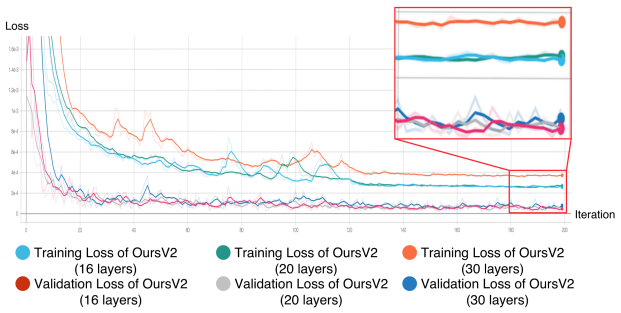

Figure 7: Training and validation loss graph of OursV2 models.

As states in the neural network architectures section, OursV2 has diferent versions which has a depth of 16, 20 and 30 layers. These models are made of same building blocks stacked different number of times. Thus, by again just looking at the training and validation loss values some assumptions about the effects of the depth to the accuracy of the model can be made. At figure 7 training and validation loss graph of OursV2 models is given. Although further investigation of other metrics are needed, it can be estimated that increasing the depth of the model does slightly increase the accuracy.

Also, by looking to the output of the models which is the denoised images, some more estimations can be made. To elaborate on that, sample test results for all models can be found in the figure 8 and figure 9. As it can be seen from these figures, there are minor anomalies in the results of OursV1 models. As stated before, this issue can be caused by the max pooling and max unpooling operations which causes loss of image details. [8]

However, most of the time looking to the output of the model which is the denoised images is not practical as it will be hard to distinguish the difference. Also, altough the loss graph gives a lot of information about the model, the information it contains is not sufficient to make a solid comparison between models. Thus, while comparing denoising models a more reliable way is used in the literature which is calculating average PSNR and SSIM scores and making the comparison based on these metrics. At table 3, average PSNR and SSIM scores for each model is given in order to make a better comparison.

As it can be seen from the table 3, OursV2 models have the best PSNR and SSIM scores. Thus, it can be concluded that within all models introduced in the neural network architectures section, OursV2 models are performing better than other models. In the next section only OursV2 models will be compared with the state-of-the-art methods.

Table 3: Average PSNR and SSIM scores.

Model Name	Average PSNR	Average SSIM
OursV0	33.97	0.904
OursV0 (w skip connections)	33.09	0.902
OursV1	29.74	0.213
OursV1 (w interpolation)	29.86	0.292
OursV1 (w maxpooling)	28.70	0.202
OursV2 (16 layer)	39.18	0.921
OursV2 (20 layer)	37.46	0.941
OursV2 (30 layer)	39.47	0.912

Moreover, one other aspect which needs a clarification is that there is a conflict in the OursV2 16, 20 and 30 layer results. In order to better discuss the conflict, results of only these models are given in the table 4.

Table 4: OursV2s results.

Method Name	Average PSNR	Average SSIM
OursV2 (30 layer)	39.47	0.912
OursV2 (16 layer)	39.18	0.921
OursV2 (20 layer)	37.46	0.941

As it can be seen from the table, there is a negative correlation between PSNR and SSIM scores. Cause of this matter might be sensitivity of SSIM index to the distortion in the image while PSNR is sensitive to the intensity difference. [9] With respect to this information it can be stated that the proposed models are slightly increasing distortion of the image while reducing the intensity diferance. In the literature, to resolve this issue more sophisticated loss functions are used instead of mean square error.

5.1. Related Work

With the rise of the neural networks, extensive studies have been conducted to develop various image denoising methods. Thus, there are many diferent approaches to the denoising task. Within this project, mainly stacked autoencoders are studied in detail. However, there are many other methods which perform well in denoising task.

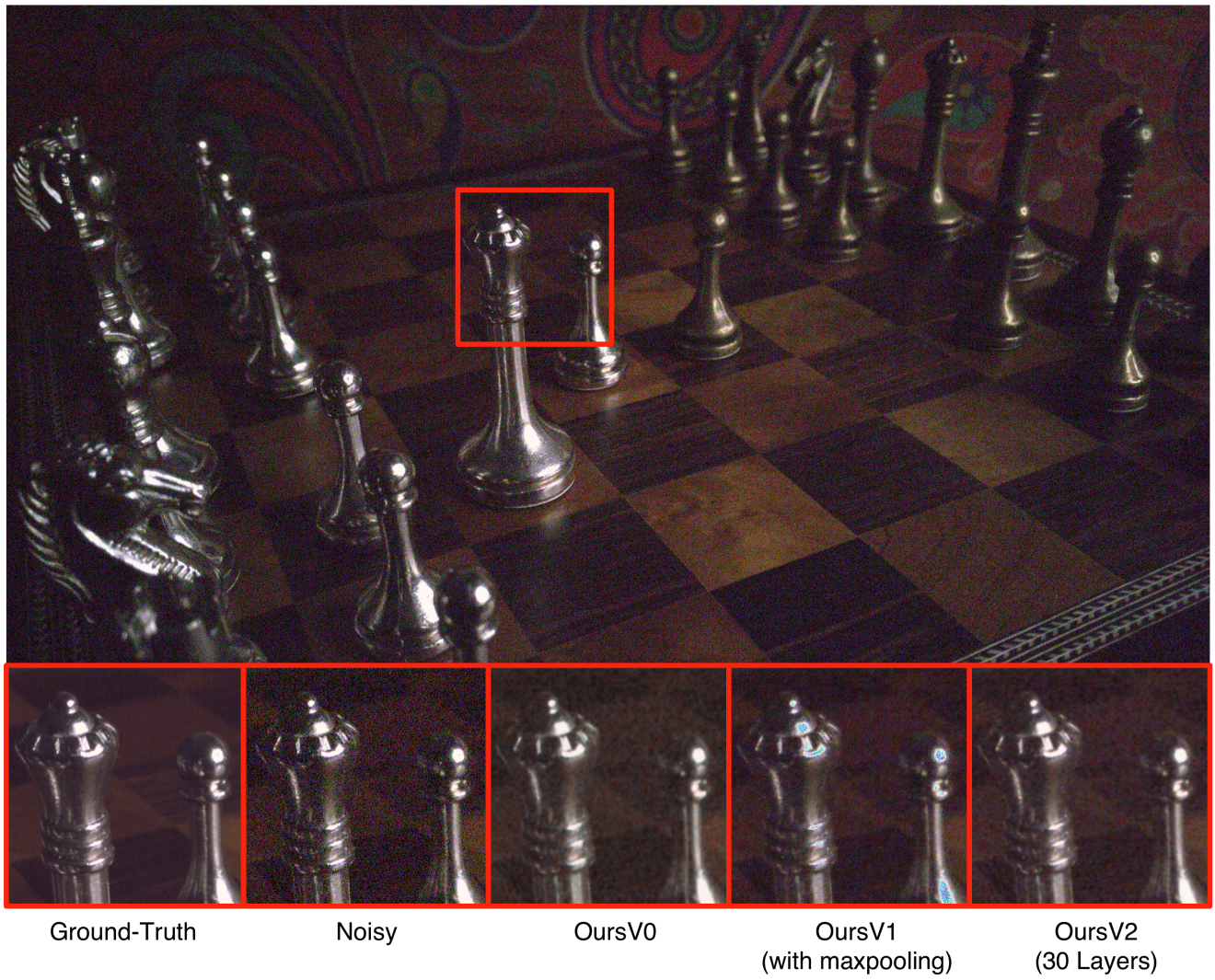

Figure 8: Sample denoising results for all 3 methods.

At this section, benchmark results wil be introduced. Following that, some of the widely used denoising methods in the literature will be briefly explained.

Smartphone image denoising dataset does not only provide a high-quality dataset but is also establishes a testbed to compare denoising methods. [2] An extensive benchmark table is provided in the smartphone image denoising dataset website. At the table 5, benchmarking data from SIDD combined with the benchmarking data of OursV2s is given.

As it can be seen from the table 5, denoising performance of OursV2 (16 layer), OursV2 (20 layer) and OursV2 (30 layer) are close to state-of-the-art methods. These results might be the outcome of using residual learning approach.

5.1.1 DN-ResNet

Residual learning led to a significant increase in performance in most computer vision tasks. Today, DN-ResNet is considered as a state-of-the-art architecture which uses residual learning combined with cascade training and edge-aware loss function. [11]

5.1.2 VDN

Variational Denoising Networks (VDNs) aims to construct a variational parametric approximation to the posterior of the latent variables, including the latent clean image and the noise variances, conditioned on the noisy image. [13] Today, VDNs are considered as one of the state-of-the-art image denoising methods.

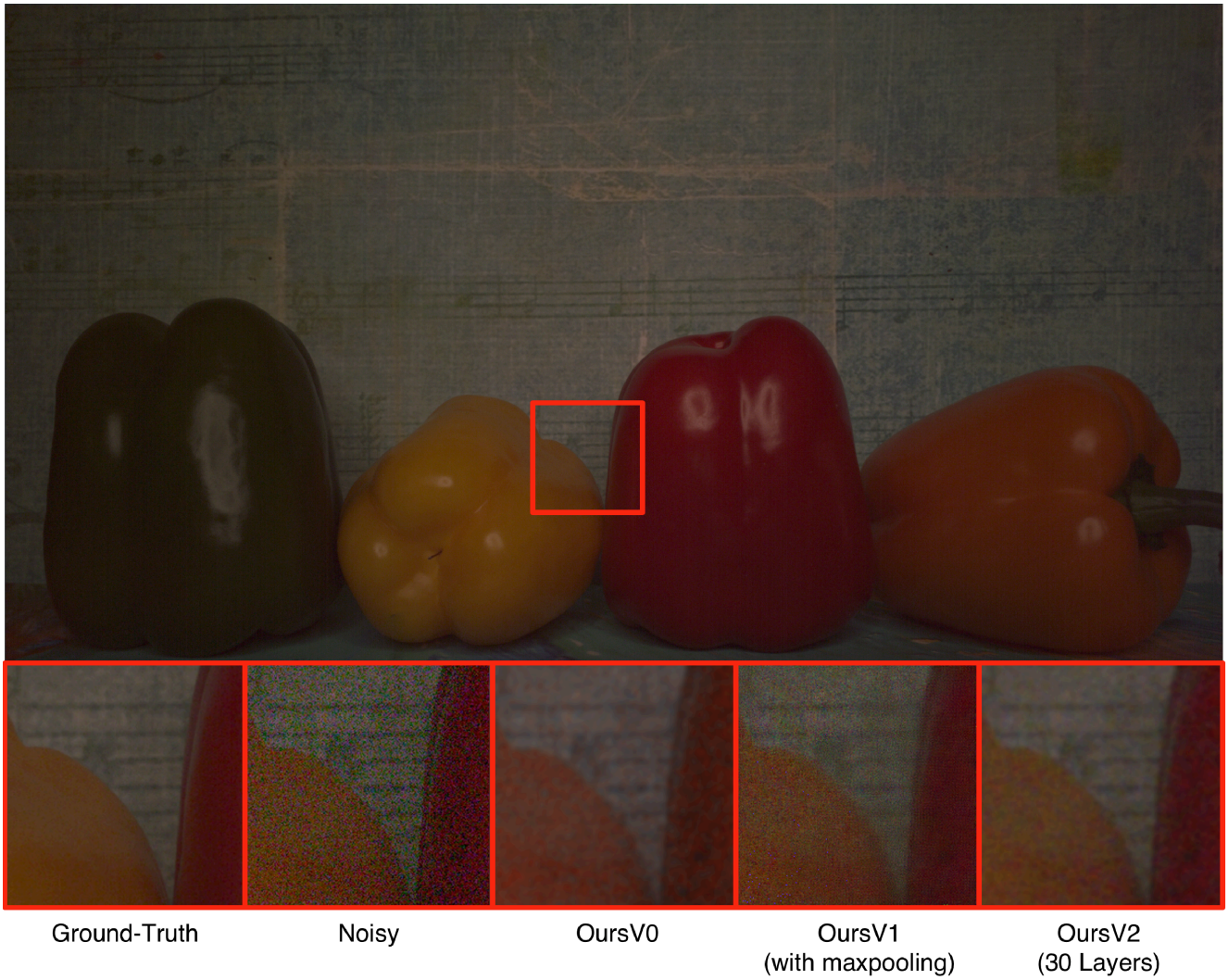

Figure 9: Sample denoising results for all 3 methods.

5.1.3 U-Net

U-Net architecture is one of the well known architectures in the literature which is used for diferent applications such as segmentation and denoising. [12] Today, it is also considered as a state-of-the-art method.

5.1.4 BoostNet

BoostNet [15] is also one of the state-of-the-art methods which is originally developed for image deblocking problem. Since image deblocking and image denoising are both similar low-level computer vision problems, BoostNet is also suitable for image denoising problem.

5.1.5 BM3D

Block-matching and 3D filtering (BM3D) is a filter based denoising method which is an altenative to the plain neural network based methods. [3] BM3D uses a specific nonlocal image modeling through a procedure termed grouping and collaborative filtering. Grouping finds mutually similar 2-D image blocks and stacks them together in 3-D arrays. Collaborative filtering produces individual estimates of all grouped blocks by filtering them. [4] BM3D is widely known in the literature and used as a baseline for image denoising.

Table 5: Benchmarking results from SIDD combined with OursV2s results.

Method Name	Average PSNR	Average SSIM
HT-MWResnet	39.80	0.959
mwresnet	39.64	0.958
OursV2 (30 layer)	39.47	0.912
VDN	39.26	0.955
OursV2 (16 layer)	39.18	0.921
UNet_D	38.88	0.952
BoostNet	38.57	0.950
OursV2 (20 layer)	37.46	0.941
BM3D	25.65	0.685
DnCNN	23.66	0.583

5.1.6 DnCNN

DnCNN is a convolutional image denoising model which adopts the residual learning formulation, and incorporate it with batch normalization for fast training and improved denoising performance. DnCNN is widely used architecture in the literature as a baseline for image denoising. [14]

6. Conclusion

In this project various deep autoencoder based neural network architectures are combined with residual learning in order to solve the denoising task which is one of the known low-level computer vision problems. Results of the study shows that when autoencoder architecture is combined with residual learning in order to avoid degradation problem, resulting architecture can achieve similar performance with state-of-the-art image denoising methods.

6.1. Impacts of the Project

6.1.1 Social Impacts

Nowadays almost all of us have some sort of mobile device which can take photographs. Improving those photos can maximize what we get out from those devices.

6.1.2 Ethical Issues

This project is for experimental use only the work done might not be production quality. Thus, using this project in mission critical systems such as autonomous driving systems will not be ethical and should not be done.

7. Acknowledgements

I would like to thank Assist. Prof. Dr. M. Furkan Kıraç for his excellent support throughout the whole semester.

References

[1] Abdelrahman Abdelhamed, Stephen Lin, and Michael S. Brown. A high-quality denoising dataset for smartphone cameras. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.

[2] Abdelrahman Abdelhamed, Radu Timofte, Michael S. Brown, et al. Ntire 2019 challenge on real image denoising: Methods and results. In The IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), June 2019.

[3] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with bm3d? In 2012 IEEE Conference on Computer Vision and Pattern Recognition, pages 2392–2399, 2012.

[4] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. BM3D Image Denoising with Shape-Adaptive Principal Component Analysis. In Rémi Gribonval, editor, SPARS'09 - Signal Processing with Adaptive Sparse Structured Representations, Saint Malo, France, Apr. 2009. Inria Rennes - Bretagne Atlantique.

[5] Simon Haykin. Neural Networks: A Comprehensive Foundation. Prentice Hall PTR, USA, 1st edition, 1994.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[7] Jayanth Koushik. Understanding convolutional neural networks. 05 2016.

[8] Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. Image restoration using convolutional auto-encoders with symmetric skip connections. 06 2016.

[9] Peter Ndajah, Hisakazu Kikuchi, Masahiro Yukawa, Hidenori Watanabe, and Shogo Muramatsu. An investigation on the quality of denoised images. International Journal of Circuits Systems and Signal Processing, 5:423–, 07 2011.

[10] Andrew Ng. Sparse autoencoder lecture notes. 06 2011.

[11] Haoyu Ren, Mostafa El-Khamy, and Jungwon Lee. Dn-resnet: Efficient deep residual network for image denoising, 2018.

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, pages 234–241, Cham, 2015. Springer International Publishing.

[13] Zongsheng Yue, Hongwei Yong, Qian Zhao, Lei Zhang, and Deyu Meng. Variational denoising network: Toward blind noise modeling and removal, 2019.

[14] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. IEEE Transactions on Image Processing, 26(7):3142–3155, Jul 2017.

[15] C. Zhao, J. Zhang, R. Wang, and W. Gao. Boostnet: A structured deep recursive network to boost image deblocking. In 2018 IEEE Visual Communications and Image Processing (VCIP), pages 1–4, 2018.