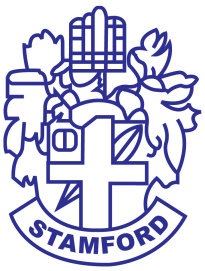
Self2self with Cutout: A Self-supervised Single Image Denoising Schema

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Bachelor in Computer Science & Engineering

by

Angraj Md. Sargia Showrav
ID: CSE 06607820
&
Sajib Kotal
ID: CSE 06607811

Supervised by: Dr. Md. Tauhid Bin Iqbal
Assistant Professor



Department of Computer Science and Engineering
STAMFORD UNIVERSITY BANGLADESH

September 2022

Abstract

Despite being challenging, research on single image based denoising is enjoying a recent upsurge due to the impressive and dominating performance of deep networks. Most of the self-supervised single image denoising methods need noisy training pair to learn the denoising function, which, however, often undergoes identity mapping and overfitting while training. A recent work, Self2Self, proposed a Bernoulli-dropout based de-noising schema that removes random pixel information to escape identity mapping. Nonetheless, real camera noises are signal dependent, and typically poses trivial changes to the images. Hence, pixels on an area might still preserve similar contextual information even under such a pixel-dropout strategy, raising the chance of suffering identity mapping. In this work, we address this critical issue by generating the training pair by randomly masking out square regions rather than simple Bernoulli-dropping, which provides a better chance to evade the relevant contextual information. Training pair is passed to the network within a self-supervised loss to generate single predicted image at each iteration, which are then averaged to acquire the final denoised image. We evaluate our method under the presence of additive and real-world noise, and observe better performance against existing traditional and self-supervised based models. Index Terms-Image Denoising, Self-supervised Learning, Masking squared region, Real-world noise

Approval

The Thesis Report "Self2Self with Cutout: A Self-supervised Single Image Denoising Schema" submitted by ANGRAJ MD SARGIA SHOWRAV; ID: CSE06607820 & SAJIB KOTAL ; ID: CSE06607811, to the Department of Computer Science & Engineering, Stamford University Bangladesh, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science (Hons) in Computer Science & Engineering and approved as to its style and contents.

Board of Examiner's Name, Signature, and Date:

.....	.....	.....
Samia Sultana	**Mashiwat Tabassum Waishy**	**Saima Siddique Tashfia**
Date:	Date:	Date:

Supervisor's Signature and Date:

.....
Dr. Md. Tauhid Bin Iqbal

Date:

Declaration

We, hereby, declare that the work presented in this Thesis is the outcome of the investigation performed by us under the supervision of Dr. Md. Tauhid Bin Iqbal, Assistant Professor, Department of Computer Science & Engineering, Stamford University Bangladesh. We also declare that no part of this Thesis and there of has been or is being submitted elsewhere for the award of any degree.

Signature and Date:

.....
Angraj Md. Sargia Showrav
Date:

.....
Sajib Kotal
Date:

Acknowledgments

Firstly, we want to thank our honorable supervisor Dr. Md. Tauhid Bin Iqbal, Assistant professor, Stamford University Bangladesh for his continuous guidance, advice, effort and invertible suggestion throughout the research. His super-vision led us to the greater knowledge that anybody seeks. We are grateful and lucky to have him. We are grateful to Ms. Jubyrea for helping us throughout the study. We want to thank our fellow research members whom we learned many things that helped us a lot. We inspired to work from published research paper that is Y. Quan, M. Chen, T. Pang, and H. Ji, "Self2self with dropout: Learning self-supervised de-noising from single image," in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 1890–, 2020.

Table of Contents

List of Figures

List of Tables

1 Introduction

Owing to the influence of environment, transmission channel, and other factors, images are inevitably con-terminated by noise during acquisition, compression, and transmission, leading to distortion and loss of image information. With the presence of noise, possible subsequent image processing tasks, such as video processing, image analysis, and tracking, are adversely affected. Therefore, image de-noising plays an important role in modern image processing systems.

1.1 What is digital image?

A digital image is a representation of a real image as a set of numbers that can be stored and handled by a digital computer. In order to translate the image into numbers, it is divided into small areas called pixels. For each pixel, the imaging device records a number, or a small set of numbers, that describe some property of this pixel, such as its brightness (the intensity of the light) or its color. The numbers are arranged in an array of rows and columns that correspond to the vertical and horizontal positions of the pixels in the image. Digital images have several basic characteristics. One is the type of the image. For example, a black and white image records only the intensity of the light falling on the pixels. A color image can have three colors, normally RGB (Red, Green, Blue) or four colors, CMYK (Cyan, Magenta, Yellow, blacK).

1.2 What is noise in image?

Noise in an image is the presence of artifacts that do not originate from the original scene content. Generally speaking, noise is a statistical variation of a measurement created by a random process. In imaging, noise emerges as an artifact in the image that appears as a grainy structure covering the image. Noise is always presents in digital images during image acquisition, coding, transmission, and processing steps. Noise can have different

forms and appearances within an image and is, in most cases, an unwanted or disturbing artifact that reduces the subjective image quality.

Figure 1.1: A clean image

Figure 1.2: A noisy image

1.3 Types of noise & their Causes

Image noise can range from almost imperceptible specks on a digital photograph taken in good light, to optical and radioastronomical images that are almost entirely noise, from which a small amount of information can be derived by sophisticated processing. Such a noise level would be unacceptable in a photograph since it would be impossible even to determine the subject.

1.3.1 Gaussian noise

Principal sources of Gaussian noise in digital images arise during acquisition. The sensor has inherent noise due to the level of illumination and its own temperature, and the electronic circuits connected to the sensor inject their own share of electronic circuit noise. A typical model of image noise is Gaussian, additive, independent at each pixel, and independent of the signal intensity.

Figure 1.3: Gaussian noise

1.3.2 Salt-and-pepper noise

Fat-tail distributed or "impulsive" noise is sometimes called salt-and-pepper noise or spike noise. An image containing salt-and-pepper noise will have dark pixels in bright regions and bright pixels in dark regions. This type of noise can be caused by analog-to-digital converter errors, bit errors in transmission, etc.

Figure 1.4: Salt and pepper noise

1.3.3 Shot noise

The dominant noise in the brighter parts of an image from an image sensor is typically that caused by statistical quantum fluctuations, that is, variation in the number of photons sensed at a given exposure level. This noise is known as photon shot noise.

1.3.4 Quantization noise

The noise caused by quantizing the pixels of a sensed image to a number of discrete levels is known as quantization noise. It has an approximately uniform distribution.

1.3.5 Periodic noise

A common source of periodic noise in an image is from electrical interference during the image capturing process. An image affected by periodic noise will look like a repeating pattern has been added on top of the original image. In the frequency domain this type of noise can be seen as discrete spikes.

1.3.6 In Digital Cameras

In low light, correct exposure requires the use of slow shutter speed (i.e. long exposure time) or an opened aperture (lower f-number), or both, to increase the amount of light (photons) captured which in turn reduces the impact of shot noise. If the limits of shutter (motion) and aperture (depth of field) have been reached and the resulting image is still not bright enough, then higher gain (ISO sensitivity) should be used to reduce read noise. On most cameras, slower shutter speeds lead to increased salt-and-pepper noise due to photodiode leakage currents.

1.3.7 Read noise

In digital camera photography, the incoming photons are converted to a voltage. This voltage then passes through the signal processing chain of the digital camera and is digitized by an analog to digital converter. Any voltage fluctuations in the signal processing chain, that contribute to a deviation of analog to digital units, from the ideal value proportional to the photon count, is called read noise.

1.3.8 Effects of sensor size

The size of the image sensor, or effective light collection area per pixel sensor, is the largest determinant of signal levels that determine signal-to-noise ratio and hence apparent noise levels, assuming the aperture area is proportional to sensor area, or that the f-number or focal-plane illuminance is held constant. That is, for a constant f-number, the sensitivity of an imager scales roughly with the sensor area, so larger sensors typically create lower noise images than smaller sensors. In the case of images bright enough to be in the shot noise limited regime, when the image is scaled to the same size on screen, or printed at the same size, the pixel count makes little difference to perceptible noise levels – the noise depends primarily on sensor area, not how this area is divided into pixels. For images at lower signal levels (higher ISO settings), where read noise (noise floor) is significant,

more pixels within a given sensor area will make the image noisier if the per pixel read noise is the same.

1.3.9 Sensor fill factor

The image sensor has individual photosites to collect light from a given area. Not all areas of the sensor are used to collect light, due to other circuitry. A higher fill factor of a sensor causes more light to be collected, allowing for better ISO performance based on sensor size.

1.3.10 Sensor heat

Temperature can also have an effect on the amount of noise produced by an image sensor due to leakage. With this in mind, it is known that DSLRs will produce more noise during summer than in winter.

1.4 Objective of Image denoising

In the modern era, Deep learning Continues to contribute to Computer Vision, image processing and AI. Among them, Image denoising is very Basic topic [1], [2]. The purpose of noise reduction is to decrease the noise in natural images while minimizing the loss of original features and improving the signal-to-noise ratio (SNR). The major challenges for image denoising are as follows:

- flat areas should be smooth,
- edges should be protected without blurring,
- textures should be preserved, and
- new artifacts should not be generated.

Generally noisy image is defined as follows-

$$y = x + n \tag{1.1}$$

Where y is a noisy image, x is a clean image and n is noise said to be pixel independent, and mean-zero. Normally denoising or reconstruction of noisy image is done by neural

network which learning the mapping from many pairs of clean and noisy image, and solve the following equation for $\mathscr{L}$ to be the loss the function and f be the denoiser. which is trained by the loss function. If the loss function is $\mathscr{L}$ then the equation holds-

$$min \sum_{\forall i} \mathscr{L}(f(x^i), y^i). \tag{1.2}$$

However, in real world, having such pairs are difficult, costly, and in some cases almost impossible. Such issues are usually tackled by artificially adding white Gaussian, Poisson noise, or mixture of Gaussian-Poisson noise. Some recent works even use generative modeling to generate noise patches for creating large data-set of noisy, clean pair to train neural network [3], [4], [5]. Although these models along with other traditional deep denoisers [6], [7], [8], achieve impressive performance on artificially added noise, but there is still. enough room to improve their performance on real camera images [9], [10].

1.5 Single Image Denoising

The standard denoising methods [3], [4], [5], [6] uses noisy, clean pairs to feed the neural network. But in real scenario this case is too costly & almost impossible. With that in mind noise2noise[11] was introduced that uses pairs of images from the same noisy distribution. This method reduces the dependency of having clean images as pair. We need many pairs of noisy image under certain noisy distribution to learn the network, that is still difficult in practical implementation. That's when researcher' s thought about self-supervised learning where the network will learn to predict the clean image by the noisy image itself. Noise2void [12], Noise2self [13], Self2self [14] showed that, we can construct a clean image from a noisy image itself. With the advantage of no need of any noisy/clean or noisy/noisy pair to deal with. Self2self [14] introduced the Bernoulli-dropout method that randomly dropout pixels over Bernoulli sampled instance. That performs state-of-the-art performance over current image denoising methods. Thus said, in this method pixels are dropping at the input side and are predicted by the averaging of neighbouring pixels. Now for additive noise where the noise is independent to signal this theory stands as there should not be any correlation between neighbouring pixels. But In real image, noise is dependent & correlation exists among neighbouring pixels. Here is some signal distribution of noisy & clean real world images shown in Figure 1.5.

Signal distribution of random noisy & clean real world images from polyU dataset,

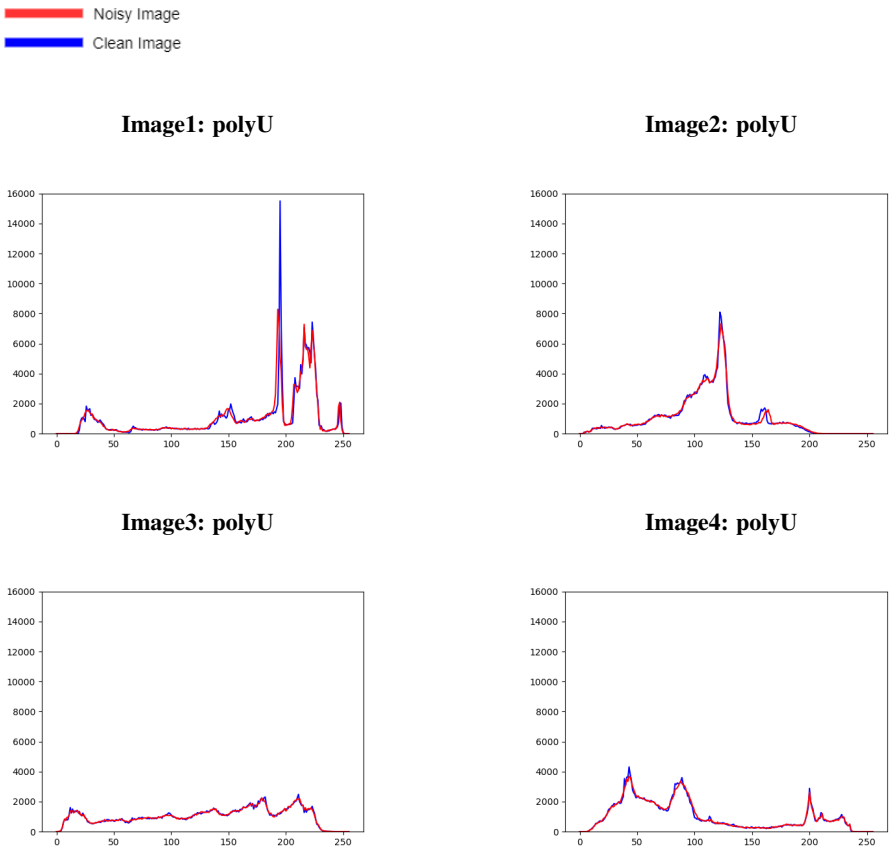


Figure 1.5: Signal distribution of noisy & clean real world images

1.6 Motivation

The above data shows that in real world image, signal distribution of noisy images & clean images are almost same. So we can say noise are distributed according to true pixel intensity. Hence real noise are pixel dependent. So we can say that surrounding pixels are correlated & after dropping a pixel they still contains contextual information of the dropped pixel. In that case, even after dropping pixels there is still very much possibility of occurring identity mapping and overfitting.

Cutout [15] solves this inevitable problem that we faces by dropping random square regions of an image. This removes the possibility of noise dependency, correlation & escapes further identity mapping & overfitting.

Motivated by the advantages of cutout [15], we tend to incorporate the idea of on our single image denoising schema. During training, instead of dropping pixels to generate pairs of images as done in self2self [14], we generate pairs by randomly masking out square regions. The training pair is passed to a denoising network and self-supervised loss is applied on the pairs. The process is iterated multiple times and the outputs from each iteration are averaged to acquire the final denoised image. Note that external horizontal/vertical flipping of the input and node dropping from fc layers are also employed to ensure further regularization. Thus, the training scheme, at a whole, helps the network avoiding overfitting and identity mapping while ensure a reliable performance. We have tested our denoising schema in both additive and real-world noise, and observed better performance than other existing self-supervised and traditional-learning based methods.

The proposed denoising scheme: Our input to the network is a noisy image (a), (b) is the noisy input to the network after masking n number of $m \times m$ square regions. In (c) we represent the proposed denoising network containing encoder and decoder blocks. Final output is the denoised image (d), which can be better viewed via zooming in. For ease, area under red box is shown after zooming in for a clear view.

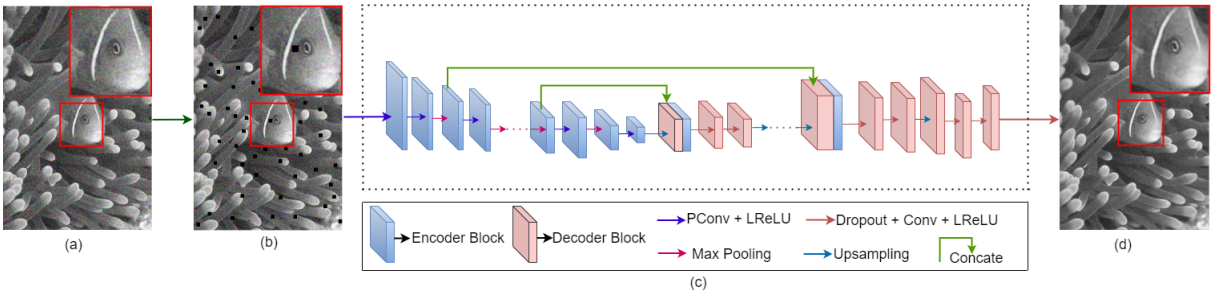


Figure 1.6: Proposed denoising scheme

2 Literature Review

In this section we have elaborated background working methods of the denoising imaging era.

2.1 Background study

In a recent work, Noise2noise [11], authors demonstrate a new approach to reconstruct image from same underlying distribution without the need of clean image. The network is trained on pair of noisy input under certain distributional assumptions, where the network learns to predict the clean image implying a self-supervised loss. Some works have also been done on the direction where no target image is needed, rather network learns to construct the clean image from noisy image itself [12], [13], [14]. However, such single image based self-supervised learning strategies may at times experience identity mapping and over-fitting issues, as pointed in [14, 15]. To address this, authors of [14] proposed a dropout-based model, where the network is trained on the pair of Bernoulli sampled dropout-driven instances introducing random variation to the network to reconstruct the clean image. One major purpose of such random pixel dropping is to remove pixel information to evade the network from identity mapping. Although image denoising is performed under the basic assumption that the noise to be independent; but real camera noise is signal-dependent and often observed to bring trivial changes on input image [10]. Hence, pixels on an area can still be co-related even under the presence of such subtle noise. Therefore, dropping a pixel from one area does not totally remove its contextual information, rather still preserves the self-similarity to some extent. A simple regularization technique [15] was prescribed to address this issue of dropout, where authors proposed to randomly mask out square regions of input image, and observed improved performance in terms of robustness and accuracy. Such an approach can be advantageous since masking out a square area instead of a single pixel preserves better chance of removing contextual information and thus, evades identity-mapping.

2.1.1 Noise2Noise

With the difficulty of having many pairs of noisy & clean observations. Noise2noise [11] authors proposed a denoising method where network is learned by only looking at the noisy observations. Maintaining the same methodology of training with image pairs. Thus says, we can often learn to turn bad images into good images by only looking at bad images- as if we were using clean examples. That implies we achieve a clean target by averaging many noisy targets of same distribution. That gives us the same minimizing formula (1.2) as we have known. Thus says a long, noisy free exposure is the average of short, independent, noisy exposure. Further we require neither an explicit statistical likelihood model of the corruption nor an image prior and instead learn these indirectly from the training data. That gives us the same minimization task like equation (1.2). Where both the inputs and the targets are drawn from a corrupted distribution, that not necessarily need to be the same. But mapping with same underlying distribution gives us another problem of Identity mapping and overfitting. As it can output same noisy target. Also having different observation of same underlying distribution is also costly. To overcome this issues sel2self authors introduced a dropout technique that only requires a single noisy image.

2.1.2 Self2self with Bernoulli Dropout

Traditional works on de-noising introduced us methods that allow training a denoising network on the set of external noisy images only. Where self2self [14] proposes a self-supervised learning method which only uses the input noisy image itself for training.

$$f\theta(\cdot) : y \rightarrow x$$

It shows how to training a denoising NN using only the input noisy image y itself. This scheme uses a self-prediction loss defined on the pairs of Bernoulli sampled instances of the input image. A Bernoulli sampled instance by of an image y with probability p is defined by.

$$\hat{y}[k] = \begin{cases} y[k], with\ probability\ p \\ 0. with\ probability\ 1-p \end{cases}$$

By dropping out random pixel of a noisy image y using Bernoulli instance and creating many independent instances from the image which are different from y, yet covering most

of its information. This generates a set of Bernoulli sample instance. The network is then trained on the pair of Bernoulli sampled dropout-driven instances introducing random variation to the network to reconstruct the clean image. This hence evade identity mapping because of random pixel dropping also removes pixel information.

2.1.3 Cutout

Cutout [15] is a simple regularization technique for convolutional neural networks that involves removing contiguous sections of input images, effectively augmenting the dataset with partially occluded versions of existing samples. This technique can be interpreted as an extension of dropout in input space, but with a spatial prior applied, much in the same way that CNNs apply a spatial prior to achieve improved performance over feed-forward networks on image data Cutout units are dropped at the input stage of the network rather than in the intermediate layers. This approach has the effect that visual features, including objects that are removed from the input image, are correspondingly removed from all subsequent feature maps. Other dropout variants generally consider each feature map individually, and as a result, features that are randomly removed from one feature map may still be present in others. These inconsistencies produce a noisy representation of the input image, thereby forcing the network to become more robust to noisy inputs. In this sense, cutout is much closer to data augmentation than dropout, as it is not creating noise, but instead generating images that appear novel to the network.

Randomly contextual information removed from an image

Figure 2.1: Cutout image

3 Methodology

In this section, we will discuss the details of our training scheme and denoising strategy. The overall training procedure is depicted at Figure 1.6, where input to the network is a noisy image (a). First, we mask out random squared areas of the given image (b), and pass it through a denoising network (c) in order to generate one prediction. The process is iterated many times, and outputs from each iteration are averaged to generate the final denoised image (d).

3.1 Keywords

3.1.1 Identity mapping

An Identity Map is a function that always returns the value that was used as its argument, unchanged. That is, when f is the identity function, the equality $f(x) = x$ is true for all values of x to which f can be applied. If a network is identity mapping, then it's said that the network is returning the values that it has taken as input. Hence the network is not learning anything from the input.

$$\arg\min\ L(f(x) - x) = identity \tag{3.1}$$

$$\begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} - \begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Figure 3.1: Identity mapping

3.1.2 Overfitting

Large neural nets trained on relatively small datasets can overfit the training data. Thus says, overfitting occurs when a statistical model fits exactly against its training data. When this happens, the algorithm unfortunately cannot perform accurately against unseen data, defeating its purpose. Generalization of a model to new data is ultimately what allows us to use machine learning algorithms every day to make predictions and classify data. Again, when the model trains for too long on sample data or when the model is too complex, it can start to learn the noise or irrelevant information, within the dataset. When the model memorizes the noise and fits too closely to the training set, the model becomes overfitted, and it is unable to generalize well to new data.

3.1.3 Dropout

Dropout is a regularization method that approximates training a large number of neural networks with different architectures in parallel. During training, some number of layer outputs are randomly ignored or "dropped out". This has the effect of making the layer look-like and be treated-like a layer with a different number of nodes and connectivity to the prior layer. In effect, each update to a layer during training is performed with a different "view" of the configured layer. Dropout has the effect of making the training process noisy, forcing nodes within a layer to probabilistically take on more or less responsibility for the inputs. This conceptualization suggests that perhaps dropout breaks-up situations where network layers co-adapt to correct mistakes from prior layers, in turn making the model more robust. Dropout simulates a sparse activation from a given layer, which interestingly, in turn, encourages the network to actually learn a sparse representation as a side-effect. As such, it may be used as an alternative to activity regularization for encouraging sparse representations in autoencoder [16] models. Because the outputs of a layer under dropout are randomly subsampled, it has the effect of reducing the capacity or thinning the network during training. As such, a wider network, e.g., more nodes, may be required when using dropout.

3.1.4 Partial Convolution (PConv)

Partial Convolution performs the normalization of the output to adjust for the fraction of missing data. A partial convolution layer comprises masked and re-normalized convolution operation followed by a mask-update setup. In our method partial convolution is used to fill missing cutout holes.

3.1.5 *Denoising Auto-encoders & Context Encoders*

Denoising Auto-encoders [16] & Context Encoders [17] both rely on self-supervised learning to elicit useful feature representations of images. These models work by corrupting input images and requiring the network to reconstruct them using the remaining pixels as context to determine how best to fill in the blanks. Specifically, denoising auto-encoders that apply Bernoulli noise randomly erase individual pixels in the input image, while context encoders erase larger spatial regions. In order to properly fill in the missing information, the auto-encoders are forced to learn how to extract useful features from the images, rather than simply learning an identity function. As context encoders are required to fill in a larger region of the image, they are required to have a better understanding of the global content of the image, and therefore they learn higher-level features compared to denoising auto-encoders [14]. These feature representations have been demonstrated to be useful for pre-training classification, detection, and semantic segmentation models. While removing contiguous sections of the input has previously been used as an image corruption technique, like in context encoders, to our knowledge it has not previously been applied directly to the training of supervised models.

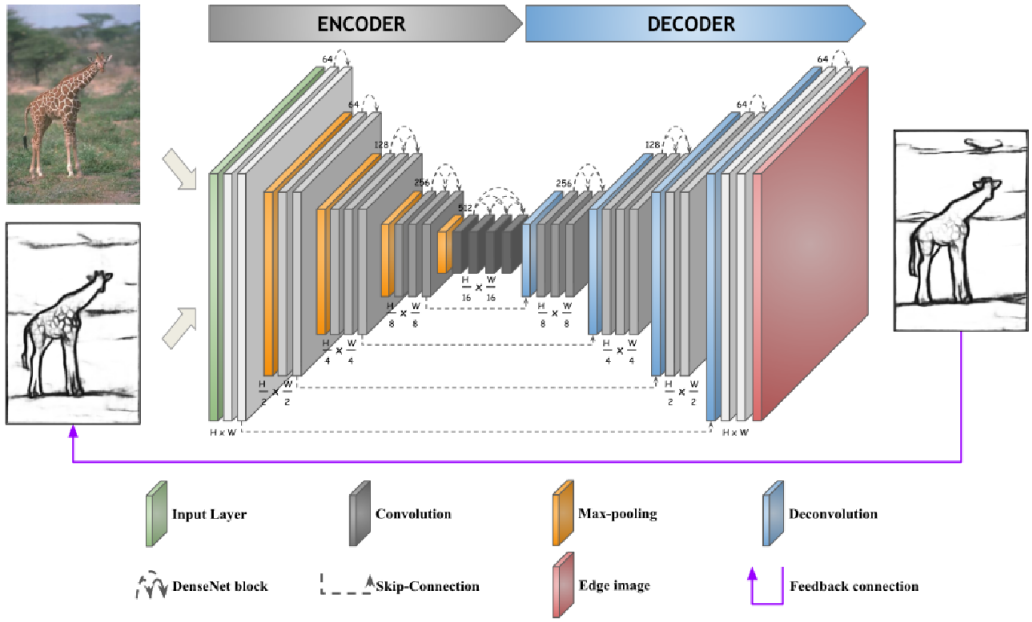


Figure 3.2: Encoder-decoder block

‘

3.2 Training Scheme

A common initial approach in single image denoising is to generate image pairs from the input noisy image [13], [14], where the pairs are different from input image yet covering most of the information. Denoising is then conducted by introducing a self supervised loss function on the pairs within a predefined denoiser network. An example of such a loss function $\mathscr{L}$ is,

$$\mathscr{L}(f(\hat{y}), \bar{y}), \tag{3.2}$$

where, $\hat{y}$ and $\bar{y}$ are the training pairs of input noisy image, y; f is a denoising function. We follow a similar strategy for the denoising purpose. To generate image pairs, we first generate a masked image by dropping square regions of input image. The masked image $\hat{y}$ is defined by:

$$\hat{y} = r \odot y, \tag{3.3}$$

where r is a binary mask of the same size of y, r contains n number of square hole of size m, and $\odot$ represents element-wise multiplication. In accordance with $\hat{y}$, we now define the second sample of the pair $\bar{y}$ as,

$$\bar{y} = (1 - r) \odot y. \tag{3.4}$$

For N training iteration we can define our pair as: $\{\hat{y}_n, \bar{y}_n\}_{n=1}^{N}$. For the denoising purpose, a self-supervised loss function is now applied to the training pairs following Eq. 3.2. The loss function $\mathscr{L}$ of Eq. 3.2 can be defined in many ways; however, in our approach we define the loss as,

$$min \sum_{1}^{N} abs \| f(\hat{y}_n) - \bar{y}_n \|. \tag{3.5}$$

The sum of total loss calculated over all training pairs in fact measures the difference over all image pixels aid preventing the network converging to any identity mapping as well. For further improvement we have use data augmentation by horizontal and vertical flipping. Note that in our approach, there is no need to create an external masked images dataset, rather we pass the copies of noisy image to the network at each iteration.

3.3 Denoising Scheme

Our network is trained with multiple instance of the given noisy image by randomly masking out areas, thus generate multiple recovered predicted images $x_1, x_2, x_3, ..., x_n$ by minimizing loss function at each iteration. Final denoised image x is the average of all predicted image as,

$$x = 1/N \sum_{n=1}^{N} x_n. \tag{3.6}$$

Due to the random dropping of squared region, our input to the network is different in each iteration. We flip our input horizontally and vertically for further augmentation while also dropping nodes in fully connected layer. All these aspects, at a whole, helps our network evading identity mapping and over-fitting while ensuring a stable performance.

The pseudo algorithm for training a denoising model is given in Algorithm 1.

Algorithm 1 Training a denoising model, all experiments used the defaults values, iteration= 200000, itr=1000, number$_{of}masked_{area} = 49, num_{of}predictions = 100$

Require: H, W, C

for $n \leftarrow iteration$ **do**
  for $i \leftarrow number_{of}masked_{area}$ **do**
    $r_n \leftarrow generate_{mask}\,([y1 : y2, x1 : x2] = 0)$
  end for
  $Sample\{\hat{y_n} = r_n \odot y; \quad \overline{y_n} = (1 - r_n) \odot y\}$
  $Train \leftarrow \{min\,\Sigma_1^N abs \parallel f_r(\hat{y_n}) - \overline{y_n} \parallel\}$
  if $k = itr$ **then**
    $Model \leftarrow evaluation$
    for $j \leftarrow number_{of}predictions$ **do**
      $sample\{predict(x_n \leftarrow Train)\}$
    end for
    $x = 1/N\,\Sigma_1^N x_n$
    $Output(x)$
  end if
end for
Return =0

3.4 Network Architecture

Our network is an encoder decoder network, where input is a noisy image of $H \times W \times C$, where the notations represent height, width, and channel respectively. First, we mask n square regions of m size from the input image using a binary mask. Then, the masked image is pass through the encoder which maps the image to an $H \times W \times 48$ feature cube along with partial convolutional layer. It is than processed by the subsequent six encoder's blocks (EBs). The first five EB successively connects a PConv layer, a leaky rectified linear unit (LReLU), and a max pooling layer with 2×2 receptive fields with a stride of 2. A PConv layer and an LReLU are connected to the last EB. Throughout all EBs we have 48 channels. A $H/32 \times W/32 \times 48$ cube size feature is the output of the encoder. Five decoder's blocks (DBs) sequentially form the decoder, each of the first four DBs consecutively connects an up-sampling layer with a scaling factor 2, a concatenation, and two standard convolutional layers with LReLUs. Convolutional layers in the first four DBs have 96 output channels, and last DB returns the feature cube back to the image of size $H \times W \times C$.

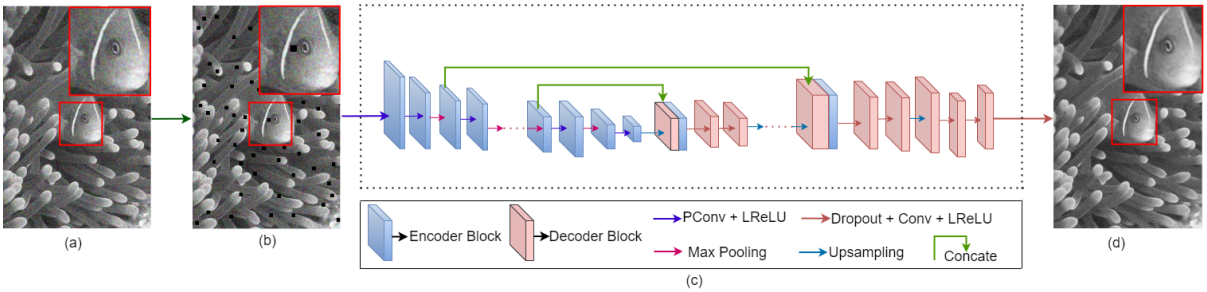


Figure 3.3: Proposed network architecture

4 Implementation & Results

The proposed method is experimented on both additive white Gaussian noise (AWGN), and real-world noise to evaluate the model's performance. Peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) [16] metrics are used to measure the performance.

4.1 AWGN & Real-World Noise

4.1.1 Additive white gaussian noise (AWGN)

Additive white Gaussian noise (AWGN) is a basic noise model used in information theory to mimic the effect of many random processes that occur in nature. The modifiers denote specific characteristics:

- Additive because it is added to any noise that might be intrinsic to the information system.
- White refers to the idea that it has uniform power across the frequency band for the information system. It is an analogy to the color white which has uniform emissions at all frequencies in the visible spectrum.
- Gaussian because it has a normal distribution in the time domain with an average time domain value of zero.

4.1.2 Real world noise

- Found in real image captured from camera or any device.
- Noise is dependent to signal.
- Cannot be modeled by an explicit distribution.
- Difficult to obtain noise-free images of training sets for deep learning.

4.2 Dataset

For real-world noisy image we have used polyU dataset. In additive white gaussian noise (AWGN) we used BSD68 and Set11 dataset.

4.2.1 PolyU

PolyU Dataset is a large dataset of real-world noisy images with reasonably obtained corresponding ground truth images. The basic idea is to capture the same and unchanged scene for many times and compute their mean image, which can be roughly taken as the ground truth image for the real-world noisy images. The rational of this strategy is that for each pixel, the noise is generated randomly larger or smaller than 0. Sampling the same pixel many times and computing the average value will approximate the truth pixel value and alleviate significantly the noise.

4.2.2 BSD68

Color BSD68 dataset for image denoising benchmarks is part of The Berkeley Segmentation Dataset and Benchmark. It is used for measuring image denoising algorithms performance. It contains 68 images. Here BSD68 is a gray version of image. In addition, two datasets of the same background information are BSD68 and CBSD68.

4.2.3 Set11

Set11 is a dataset of 11 grayscale images. It is a dataset used for image reconstruction and image compression.

4.3 PSNR & SSIM

4.3.1 PSNR

The term peak signal-to-noise ratio (PSNR) is an expression for the ratio between the maximum possible value (power) of a signal and the power of distorting noise that affects the quality of its representation. Because many signals have a very wide dynamic range, (ratio between the largest and smallest possible values of a changeable quantity) the PSNR is usually expressed in terms of the logarithmic decibel scale.

Image enhancement or improving the visual quality of a digital image can be subjective. Saying that one method provides a better-quality image could vary from person to person. For this reason, it is necessary to establish quantitative/empirical measures to compare the effects of image enhancement algorithms on image quality.

Using the same set of tests images, different image enhancement algorithms can be compared systematically to identify whether a particular algorithm produces better results. The metric under investigation is the peak-signal-to-noise ratio. If we can show that an algorithm or set of algorithms can enhance a degraded known image to more closely resemble the original, then we can more accurately conclude that it is a better algorithm. The mathematical representation of the PSNR is as follows:

$$PSNR = 20log_{10}(\frac{MAX_f}{\sqrt{MSE}}) \tag{4.1}$$

Here MAX_f is the maximum signal value that exists in our original “known to be good” image and the MSE represents the average of the squares of the ”errors” between our actual image and our noisy image.

4.3.2 SSIM

The Structural Similarity Index (SSIM) [18] is a perceptual metric that quantifies image quality degradation* caused by processing such as data compression or by losses in data transmission. It is a full reference metric that requires two image is typically compressed. It may, for example. be obtained by saving a reference image as at any quality level then reading it back in. SSIM is best known in the video industry, but has strong applications for still photography. SSIM actually measures the perceptual difference between two similar images. It cannot judge which has been subjected to additional processing as data compression. The SSIM index is calculated on various windows of an image. The measure between two windows and of common size is:

$$SSIM(x,y) = \frac{((2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2))}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{4.2}$$

- μ x the pixel sample mean of x.
- μ y the pixel sample mean of y.
- σ_x^2 the variance of x.
- σ_y^2 the variance of y.
- σ_{xy} the covariance of x and y.
- c_1= $(k_1L)^2$, c_2= $(k_2L)^2$ two variables to stabilize the division with week denominator.
- L the dynamic range of the pixel values.
- k_1=0.01 and k_2=0.03 is default.

4.4 Experimental Setup

During all our experiments, we have masked p% = 0.3 square region from the input image; size of the square is empirically set to 4×4. The number of the masked regions is calculated based on the percentage of the masked area and size of input image. For an example, if we define mask size as $m \times m$ for an $h \times w$ input image, then we will have a total of $[(h \times w \times p\%)/(m \times m)]$ masked regions. Learning rate is set to $1 \times e^{-4}$, LReLU 0.1, and the optimizer is Adam. For all Conv and PConv layers kernel size is kept 3×3, stride 1, and padding zero. Note that our model is trained with masked input and evaluated with unmasked input image.

4.5 Denoising on Real world noise

Since AWGN cannot model the real camera noise appropriately, the above experimental results may not portray the real denoising performance. Real camera noise is signal dependent and heavily transformed by the camera imaging pipeline [10]. To evaluate our performance on real world noisy image, we have used PolyU dataset [19]. The dataset contains 100 real clean/noisy pairs of coloured images. For performance evaluation we have experimented with 512×512 cropped size images. Denoising performance is compared with CBM3D [18], N2V [12], N2S [13], CDnCNN [6] and Self2Self [14] with their published result. Table 4.1 illustrates the comparison among the models, where it is clear that our model achieves better PSNR against all models. It is worthwhile to note that our model achieved better PSNR than self-supervised models like N2V, N2S and Self2Self, and also surpass traditional learning method like CDnCNN [6] that is trained with external GT information.

Table 4.1: Average PSNR(dB)/SSIM for PolyU dataset.

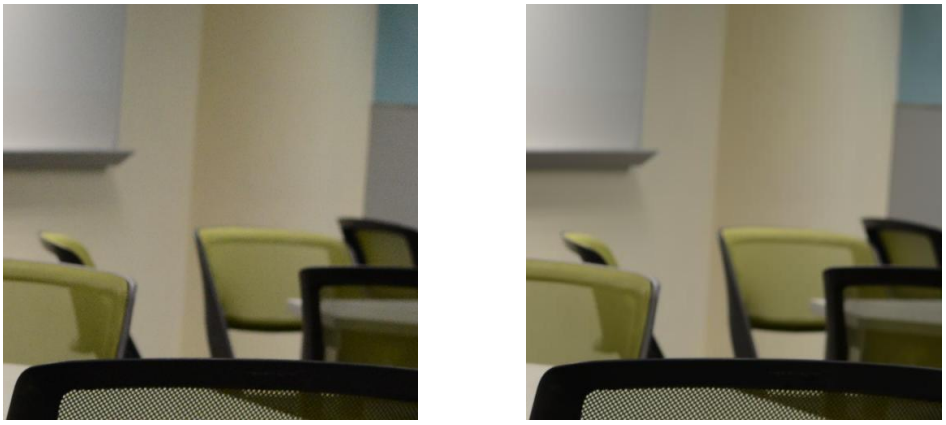
Metric	CBM3D	N2V	N2S	CDnCNN	Self2Self	Ours
PSNR	36.98	34.08	35.46	37.55	37.52	**37.57**
SSIM	0.977	0.954	0.965	**0.983**	0.980	0.942

In PolyU, CDnCNN achieves higher SSIM than ours. We like to note that CDnCNN [6] enjoys external GT information in training which might give it an edge for such a higher SSIM. However, with no such prior information, our method still provides competitive SSIM which is not negligible at all. Thus, considering the consistent performance

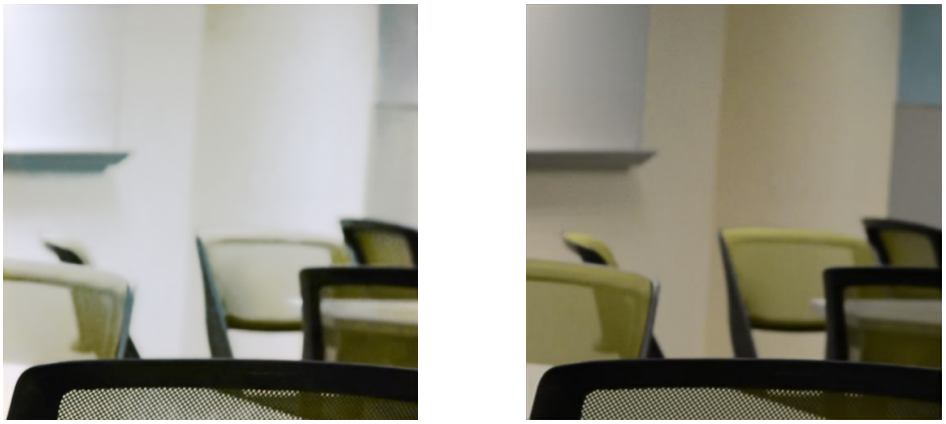
in both the additive and real-world noise, proposed method shows its overall reliability as a single image denoiser. Also our output images looks visually better performance compared to self2self. Here are some comparative visual outputs of real-world noisy images from polyU dataset in Figure 4.1,

Comparative visual outputs of real-world noisy images (a∼d) from PolyU dataset.

Noisy **GT**

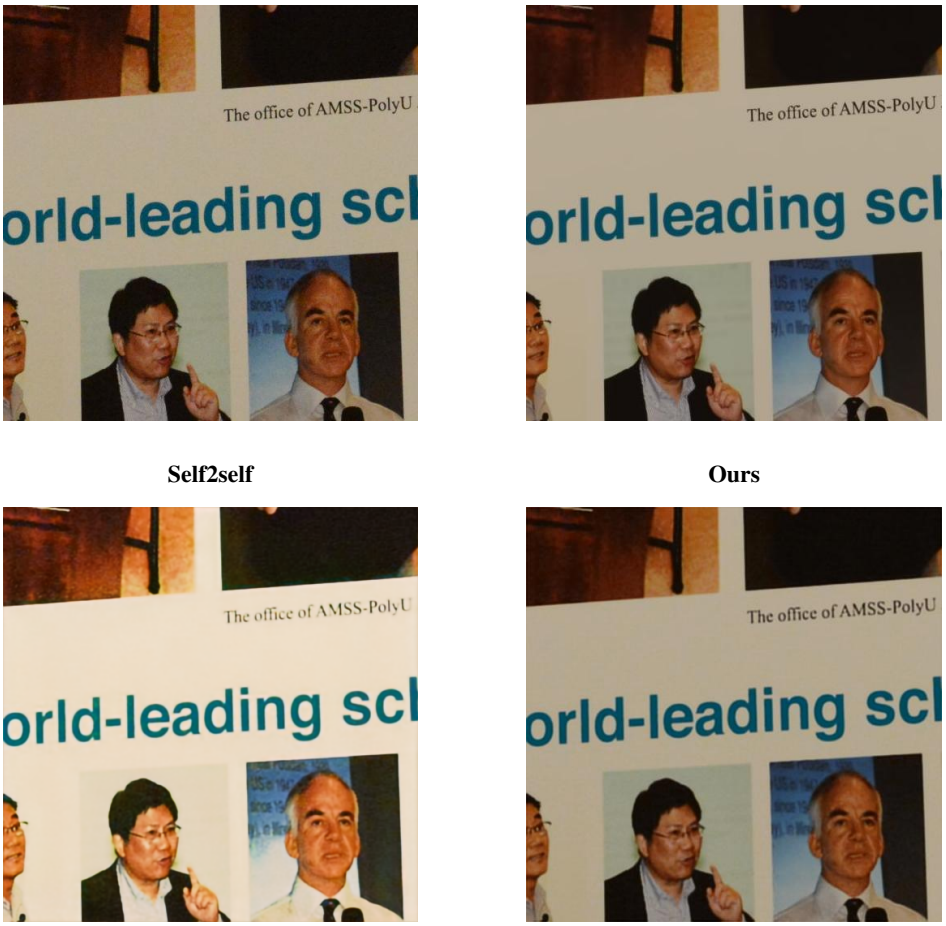


Self2self **Ours**



(a) Classroom

Noisy **GT**



Self2self **Ours**

(b) Bulletin

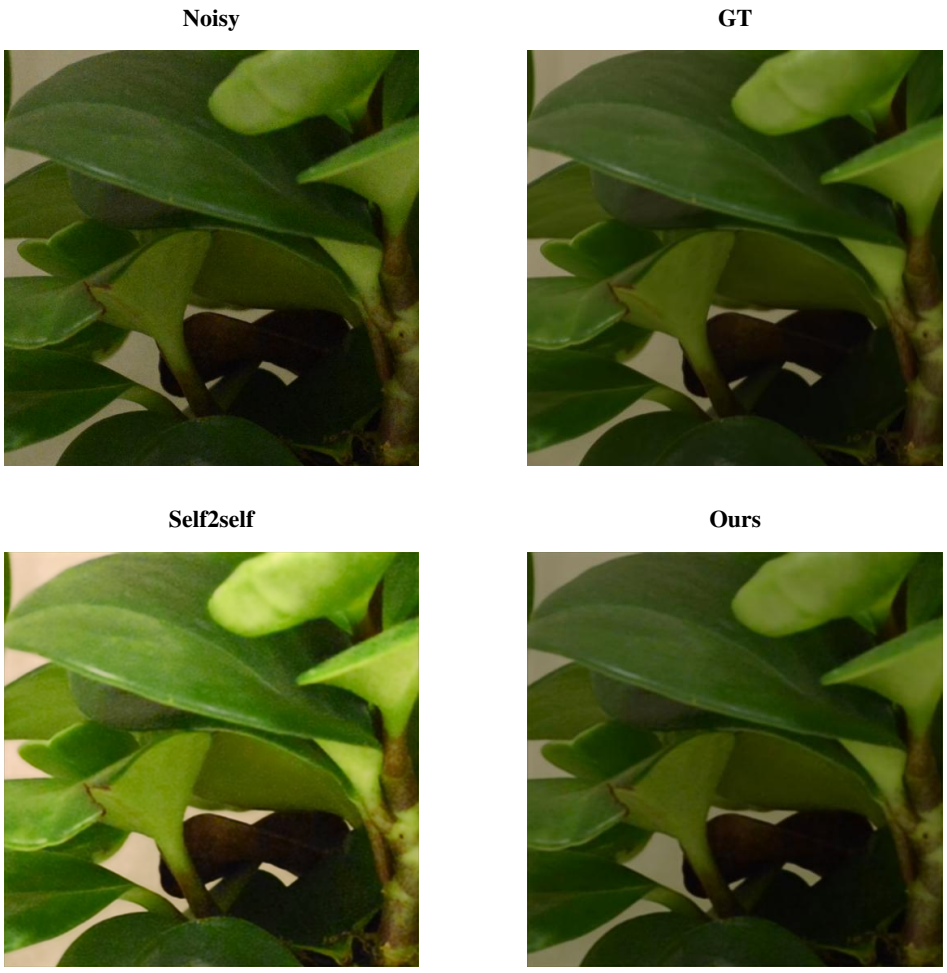
Noisy

GT

Self2self

Ours

(c) Ball



(d) Plant

Figure 4.1: Visual comparison of real-world images

4.6 Denoising on Additive noise

In this section, we have evaluated our model's performance on additive white Gaussian noise for the images of BSD68 [20] and Set-11 [7]. Among them, BSD68 (gray-scale) dataset contains 68 gray images of size 481×321, and 321×481. We have added noise level of = 25 to the images, and perform denoising with our proposed method. Denoising performance is compared with existing models like CBM3D [21], N2V [12], N2S [13], CDnCNN [6] and Self2Self [14]. Comparative denoising results for BSD68 are shown at Table 4.2, we observe that our model outperforms other methods in terms of both peak signal to noise ratio (PSNR), and structural similarity index measure (SSIM). Note that PSNR and SSIM of other given models are taken from the original published paper. SSIM is normalized to compare with the other reported results.

Table 4.2: Average PSNR(dB)/SSIM for BSD68 dataset at noise level $\sigma = 25$.

Metrics	CBM3D	N2V	N2S	CDnCNN	Self2Self	Ours
PSNR	28.56	25.34	27.19	29.14	28.70	**31.702**
SSIM	7.90	6.81	7.69	8.22	8.03	**8.35**

Table 4.3: Average PSNR(dB)/SSIM for Set-11 dataset at noise level $\sigma = 25$.

Metric	Self2Self	Ours
PSNR	28.57	**29.07**
SSIM	0.472	**0.602**

Set11 dataset contains 11 gray-scale images, and typically used for image reconstruction and image compression purpose. To test our initial hypothesis regarding masking out square region against dropout to ensure better accuracy and generalization, we compare our method in Set11 against Self2self with noise level = 25. Results at Table 4.3 shows that our method outperforms than Self2self in a significant margin in both PSNR and SSIM. Our superior performance hence in turn advocates for the efficacy of masking out square region against random pixel dropping to ensure better accuracy and robustness.

4.7 Cutout mask code

Listing 4.1: Cutout mask code

```
h = 512
w = 512
n_holes = 49
length = 4
mask = np.ones([img.shape[0],img.shape[1],img.shape
    [2]])
for n in range(n_holes):
    y = np.random.randint(h)
    x = np.random.randint(w)
    y1 = np.clip(y - length // 2, 0, h)
    y2 = np.clip(y + length // 2, 0, h)
    x1 = np.clip(x - length // 2, 0, w)
    x2 = np.clip(x + length // 2, 0, w)
    mask[y1: y2, x1: x2] = 0.
print(mask.shape)
mask = torch.from_numpy(mask)
mask = mask.expand_as(img)
print(mask, mask.shape)
img= img * mask
```

Here, h and w are corresponding to height and weight of an image, n_holes is denoted as number of holes or number of cutting area and length is represented as size of holes. Mask is defined to same size of given an input. Finally, image variable represent cutout image or output image that means this image is same size of input image but some of it's contextual information is removed. Given below the visual representation of cutout process-

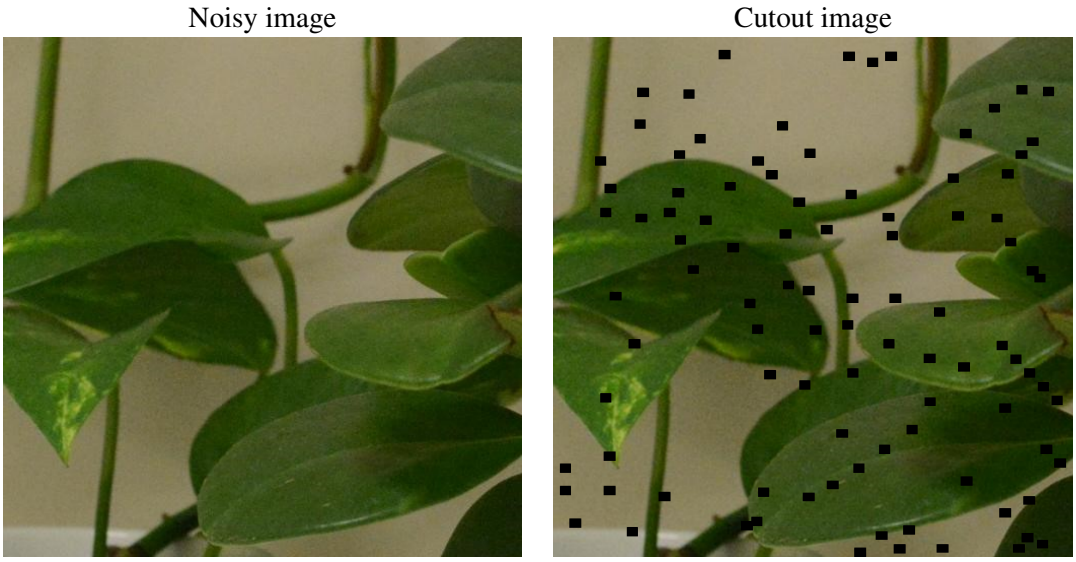


Figure 4.2: Cutting square region from noisy image

4.8 PSNR & SSIM coding implementation

Listing 4.2: PSNR code

```
original = cv2.imread("5.png")
compressed = cv2.imread("Cutout-Hole-length-
    UpdateSelf2self_avg -87000.png", 1)

def PSNR(original, compressed):
    mse = np.mean((original - compressed) ** 2)
    if (mse == 0):  # MSE is zero means no noise is
        present in the signal .
         # Therefore PSNR have no importance.
         return 100
    max_pixel = 255.0
    psnr = 20 * log10(max_pixel / sqrt(mse))
    return psnr

print(original.shape)
print(compressed.shape)


value = PSNR(original, compressed)
print(f"PSNR value is {value} dB")
```

By generating PSNR value of our predicted output, we compare our result against existing model that used this dataset. We have showed the results already in our tables. Above this code we implement according to PSNR algorithm.

Listing 4.3: SSIM code

```
from SSIM_PIL import compare_ssim
from PIL import Image
import cv2
image1 = Image.open('test001.png')
image2 = Image.open('BSD68-img1-Self2self_avg -119000.
    png')

# Compare images using OpenCL by default
value = compare_ssim(image1, image2, GPU=False)

print("SSIM",value)
```

For Calculating SSIM, we used python package Compare_ssim from SSIM_PIL.

5 Ablation Study

Here we would see how small changes in different parts of our model structure changes the performance of denoising. Also some wild concepts to justify our model more accurately as we have shown here,

5.1 Cutout Ratio

Cutout ratio determines how many pixels will be dropped in the input side with respect to cutout length and number of holes. We maintained the cutout ratio to 0.3%. That means for every 1000 pixels, 3 pixels will be dropped. That was a base assumptions still with the superior results than others. Still we wanted to see if there is any valid upgrade if we select the higher cutout ratios. By taking more smaller cutout ratio the model gets slower than before. That's why we selected bigger cutout ratio. We experimented on set9 dataset as bigger dataset is time consuming. The result given below in Table 5.1,

Table 5.1: Average PSNR(dB) of SET9 dataset over cutout ratio.

Cutout ratio	PSNR
0.1%	30.5248669
0.3%(ours)	**30.71193908**
1%	30.1746386
1.5%	29.670371
2%	30.64282337
2.5%	30.59830432
3%	30.59303389

5.2 Loss functions L1 vs L2

L_1 and L_2 are two loss functions in machine learning which are used to minimize the error.

L_1 Loss function stands for Least Absolute Deviations. Also known as LAD. L_1 Loss Function is used to minimize the error which is the sum of the all the absolute differences between the true value and the predicted value.

$$L_1 = \sum_{n=1}^{n} |y_{true} - y_{predicted}| \tag{5.1}$$

L_2 Loss function stands for Least Square Errors. Also known as LS. L_2 Loss Function is used to minimize the error which is the sum of the all the squared differences between the true value and the predicted value.

$$L_2 = \sum_{n=1}^{n} (y_{true} - y_{predicted})^2 \tag{5.2}$$

We have choosen L_1 as our loss function as it is the robust one. But we want to see how L_2 works in our model. Having other parameter same for both loss function. The comparison is showed for set9 dataset in Table 5.2,

Table 5.2: Average PSNR(dB) of SET9 dataset for loss function.

Loss function	PSNR
L_1 (ours)	**30.71193908**
L_2	30.4497

5.3 Different Cutout Shapes

Cutout shapes can also have a significant change in results. As different cutout shapes shows different parameter to calculate. In our method we have maintained our shape to square as it was proposed in [15]. Here we want to see that, whether we have implemented the optimal shape for our method. We implemented more cutout shapes like circle, ellipse & triangle on SET9 dataset to see the results in Table 5.3,

Table 5.3: Average PSNR(dB) of SET9 dataset for different cutout shapes.

Cutout shape	PSNR
Square(ours)	**30.71193908**
Circle	30.2404
Triangle	30.2615
Ellipse	30.3129

As we can see in Table 5.3, cutout of square regions gave us the higher result. So we can confirm that, this is the optimal shape for our cutout method. Here are the visual representation of different cutout shapes on set9 image at Figure 5.1,

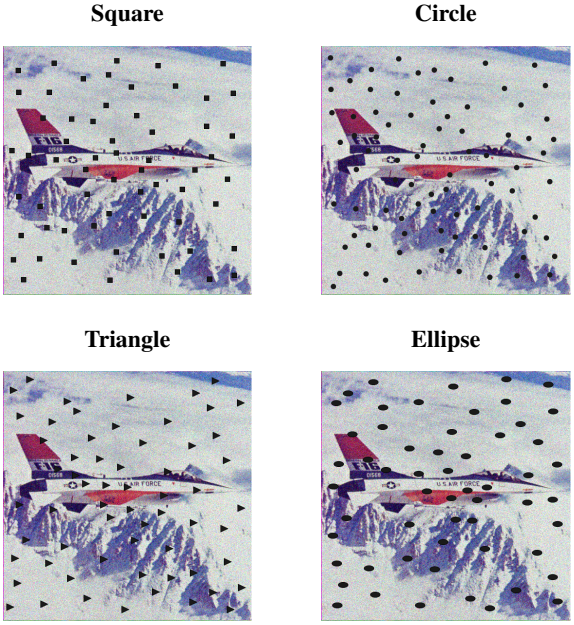


Figure 5.1: Different cutout shapes

6 Discussion & Conclusion

Our approach facilitates to train and denoises the input noisy image without the need of external data, at the same time tackle the identity mapping by masking out input image to create many samples to train the network. Consequently, vertical and horizontal flipping of input image and node-dropping at fc layer prevent the network from further overfitting. In fact, masking out square regions of image aids the network learning to reliably recover the missing regions in turn ensuring a credible denoising by averaging the network predictions. Experiments showed that denoising performance of our model is better than existing methods in both additive noise and real-world noise. We achieve superior performance than not only other self-supervised methods, but also existing traditional networks that enjoy external training data in the loop. Proposed scheme only requires single noisy acquisition to train a denoising NN, so no training data is needed. The consistent performance of our method thus makes it possible to apply denoising in heterogeneous applications where prior training data is limited but reliable denoising performance is a must. Possible such application domains are medical images (like MRI, CT), cell tracking images, solar images etc. Still there remains some unbolt questions about the optimal choice of number of masked regions, and size of the masked square area, which might be different for different dataset. We also plan to experiment our model with more real-world noisy image dataset, and domain specific areas where acquiring more image or clean target is costly, harmful or impossible; i.e., gene sequencing, images of micro-organisms, space imaging, and medical imaging.

References

[1] M. C. Motwani, M. C. Gadiya, R. C. Motwani, and F. C. Harris, "Survey of image denoising techniques," in *Proceedings of GSPX*, vol. 27, 2004, pp. 27–30.

[2] L. Fan, F. Zhang, H. Fan, and C. Zhang, "Brief review of image denoising techniques," *Visual Computing for Industry, Biomedicine, and Art*, vol. 2, no. 1, pp. 1–12, 2019.

[3] L. D. Tran, S. M. Nguyen, and M. Arai, "Gan-based noise model for denoising real images," in *Proceedings of the Asian Conference on Computer Vision*, 2020.

[4] S. Cha, T. Park, and T. Moon, "Gan2gan: Generative noise learning for blind image denoising with single noisy images," *arXiv preprint arXiv:1905.10488*, vol. 3, 2019.

[5] E. Park, Y.-J. Moon, D. Lim, and H. Lee, "De-noising sdo/hmi solar magnetograms by image translation method based on deep learning," *The Astrophysical Journal Letters*, vol. 891, no. 1, p. L4, 2020.

[6] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE transactions on image processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[7] J. Zhang and B. Ghanem, "Ista-net: Interpretable optimization-inspired deep network for image compressive sensing," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1828–1837.

[8] S. Anwar and N. Barnes, "Real image denoising with feature attention," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 3155–3164.

[9] M. Bertalmío, *Denoising of photographic images and video: fundamentals, open challenges and new trends*. Springer, 2018.

[10] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, M.-H. Yang, and L. Shao, "Cycleisp: Real image restoration via improved data synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2696–2705.

[11] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, "Noise2noise: Learning image restoration without clean data," *arXiv preprint arXiv:1803.04189*, 2018.

[12] A. Krull, T.-O. Buchholz, and F. Jug, "Noise2void-learning denoising from single noisy images," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 2129–2137.

[13] J. Batson and L. Royer, "Noise2self: Blind denoising by self-supervision," in *International Conference on Machine Learning*. PMLR, 2019, pp. 524–533.

[14] Y. Quan, M. Chen, T. Pang, and H. Ji, "Self2self with dropout: Learning self-supervised denoising from single image," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1890–1898.

[15] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[16] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." *Journal of machine learning research*, vol. 11, no. 12, 2010.

[17] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2536–2544.

[18] A. Hore and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *2010 20th international conference on pattern recognition*. IEEE, 2010, pp. 2366–2369.

[19] J. Xu, H. Li, Z. Liang, D. Zhang, and L. Zhang, "Real-world noisy image denoising: A new benchmark," *arXiv preprint arXiv:1804.02603*, 2018.

[20] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th Int'l Conf. Computer Vision*, vol. 2, July 2001, pp. 416–423.

[21] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Color image denoising via sparse 3d collaborative filtering with grouping constraint in luminance-chrominance space," in *2007 IEEE International Conference on Image Processing*, vol. 1. IEEE, 2007, pp. I–313.