International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2024: 7.101

Evaluation of Deep Learning Architectures for Image Denoising

Karthick Kumaran Ayyalluseshagiri Viswanathan

Member, IEEE

Abstract: ***Noise in the captured images has been a growing concern for various use cases not limited to mobile photography, low light imaging, drone capture, virtual reality headsets passthrough use cases etc., Recently with the advent of Apple Vision Pro and Meta Quest line of virtual reality (VR) headsets in the market, image denoising based research has got more importance to eliminate the noise from various sources, most importantly sensor noise, for improving the quality of passthrough applications. Traditional image denoising algorithms assume the noise to be Gaussian distributed but in practice the noise on the captured images can be significantly complex and so the traditional image filters fail badly for certain noise types. With the advancements in deep learning based neural networks it is now possible to remove the noise from the images so that the resulting image will be very close to the ground truth images. In this project I have implemented and evaluated two different deep learning architectures Autoencoders and U-Net for images affected with four different noise sources (Gaussian, Salt-and-Pepper, Speckle, Poisson) to denoise the images. Both the network models performance has been compared with PSNR (Peak Signal to Noise Ratio) metric and summarized the results.***

Keywords: Deep Learning, Image Denoising, Autoencoder, U-Net, Convolutional Neural Network

1. Introduction

The purpose of this project is to use the deep learning model to denoise images instead of traditional image denoising techniques such as spatial filtering and frequency domain filtering. A good denoising model should remove noise as much as possible while preserving the edges. Convolutional Neural Networks (CNNs) give better results than traditional techniques and are more computationally efficient.

While convolutional neural networks can be used for classification tasks, it is also trained for detection, recognition and generative applications. The typical use of convolutional networks is on classification tasks where the output of an image is a single class label. However, in many segmentation and generative applications the desired output includes localization in which a class label is assigned to each pixel.

While image denoising techniques require parameters to be manually set and complex optimization increases computational cost, in recent years, convolutional neural networks have proven to be more computationally efficient while producing better results. Deep neural networks like Autoencoders and U-Nets can be used for denoising images instead of the traditional image processing pipeline.

Every convolutional neural network model has a different number of layers and hyperparameters tuned based on the computational power and time needed to run the model. Different types of noises can be added to the original images such as Gaussian Noise, Poisson Noise, Salt and Pepper noise, Speckle noise for training the model.

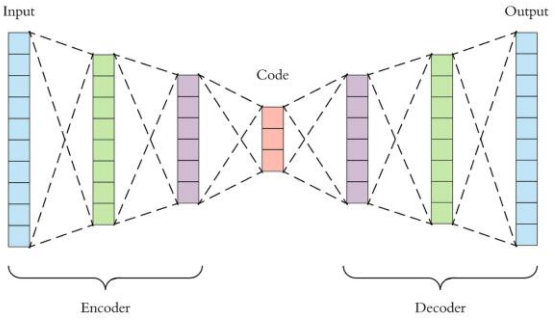
2. Prior Work

Traditionally image denoising algorithms were implemented in CPU or DSP or GPU in hand written code with one of the programming languages like C, C++, MATLAB or Assembly. These algorithms assume the noise to be Gaussian distributed or a specific noise. If the noise pattern has been changed then the algorithm needs to be reimplemented for the new noise source.

1) Deep Learning Architectures

There are different types of deep learning architectures used for a wide range of applications. Autoencoders and U-Net are identified as the perfect deep learning architecture for image denoising applications.

2) Autoencoders



Autoencoders are a specific type of feedforward neural networks where the input is the same as the output. They compress the input into a lower dimensional code and then reconstruct the output from this lower dimensional representation. The code is a compact summary also called the latent space representation.

An autoencoder consists of three components: Encoder, Decoder and the Code. The encoder compresses the input and produces the latent space representation, the decoder then reconstructs the input only using this representation. To build an autoencoder three things are required.

- An Encoder network
- Decoder network
- Loss function to compare the output with the target.

Both the encoder and the decoder networks are trained as a whole. The loss function penalizes the network for creating output that differs from the original input. By doing so the

encoder learns to preserve as much of the relevant information needed in the limitation of the latent space and discard irrelevant information, in this case, the noise. The decoder learns to take the compressed latent information and reconstruct it into a noise-free output.

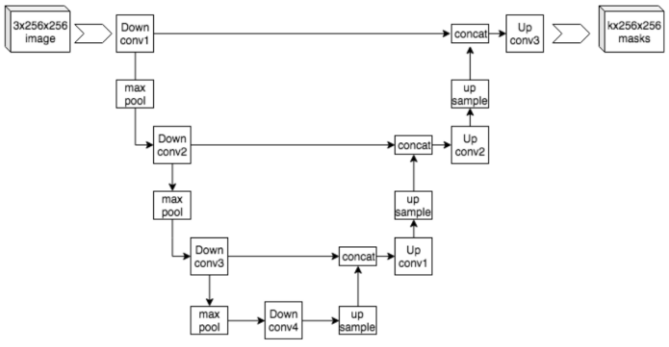
The encoder layer consists of a series of convolutional and fully connected layers. ReLU is used as the activation function. The decoder layer mirrors the structure of the encoder layer and consists of a series of fully connected and transpose convolution layers. It is the transpose convolution layers that up-samples the compressed representation in the decoder network.

3) U-Net

Autoencoders architecture has the following properties

- It takes an input and compresses the input as it goes through the layers of the encoder network. At the end of the encoder network, the input is compressed to a linear feature representation, also called, the latent space.
- The linear feature representation is upsampled by the decoder network so that at the other end of the autoencoder the result is of the same dimension as the input it received. Such architecture may be ideal for preserving the dimensionality of the output with respect to the input. But the linear compression of the input leads to a bottleneck that does not transmit all the important features.

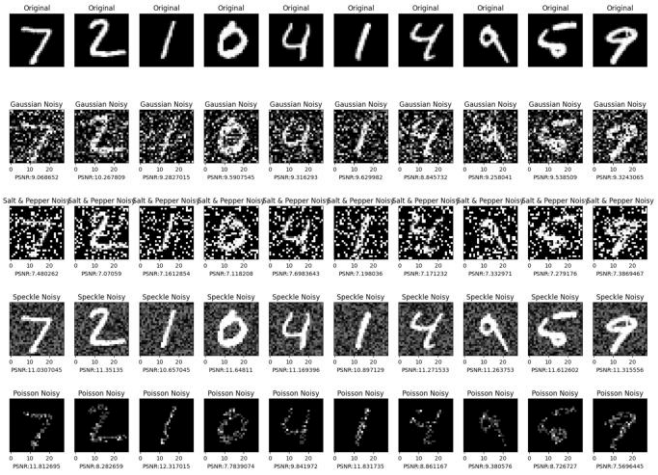
U-Net network has both the above properties listed above but it uses deconvolutional units and overcomes the bottleneck limitation by adding skip connections that allow feature representations to pass through the bottleneck. An example of U-Net architecture is shown below. It is the skip connections that make the U-Net different from Autoencoder but some machine learning engineers call U-Net as a form of Autoencoder.



U-Net Architecture

4) Image Dataset and Noise addition

Real time images from the cameras are high resolution and the dataset takes huge storage space and requires high compute capacity. It also takes days to train the network with such a huge dataset of high resolution images. So, I took the MNIST dataset for the analysis and evaluation of the two networks with four different types of noise namely Gaussian, Speckle, Salt-and-Pepper and Poisson. Images were added with each of the above mentioned noise during the pre-training phase. An example of original images and the four noises added are shown below



5) Code (Github)

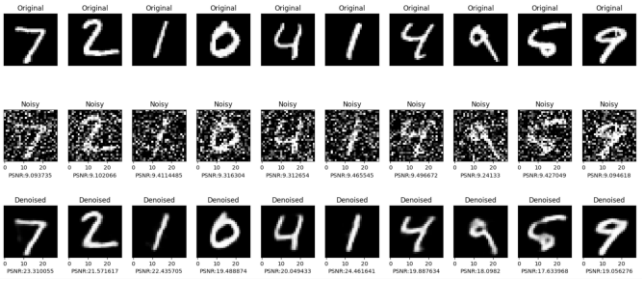
Both the autoencoder and the U-net models have been implemented and trained with different noise sources. The code is available publicly in the Github repo.

- https://github.com/asvkarthick/AdvancedMLforImaging/blob/main/Project/Adding_Noise_to_images.ipynb
- https://github.com/asvkarthick/AdvancedMLforImaging/blob/main/Project/Image_Denoising_using_Autoencoder_with_MNIST_using_different_noise_sources.ipynb
- https://github.com/asvkarthick/AdvancedMLforImaging/blob/main/Project/Image_Denoising_using_U_Net_with_different_noise_sources.ipynb

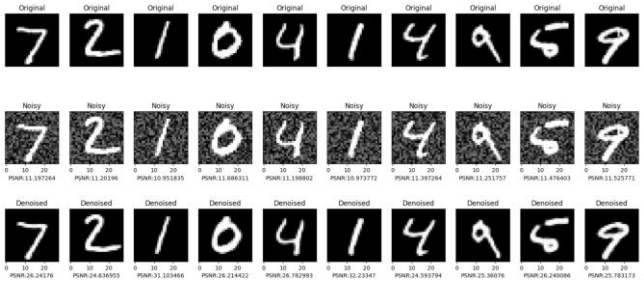
3. Evaluation of the Deep Learning Network

Both the autoencoder and the U-Net network are trained with the above mentioned datasets with added noise. The networks are supposed to denoise the noisy images in the prediction phase. In order to compare the performance of the two networks we need a metric and so PSNR (Peak Signal-to-Noise Ratio) has been used to compare the results of both the networks.

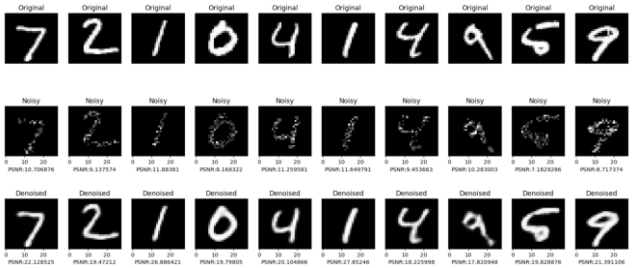
First the autoencoder network is trained with the original ground truth images and Gaussian noisy images. When the Gaussian noisy images were given for denoising (prediction) the autoencoder network removed noise from the noisy images and reconstructed the output perfectly.



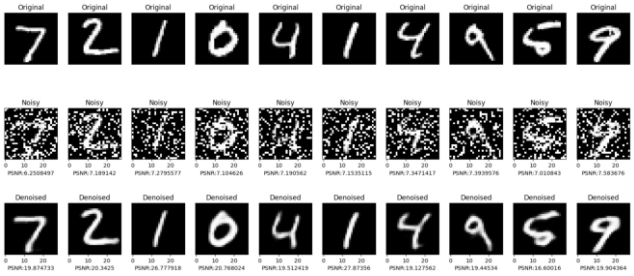
Then the autoencoder network was trained with the ground truth images and Speckle noise images from scratch. When the Speckle noisy images were given for denoising (prediction) the autoencoder network removed the Speckle noise perfectly as shown below



Then the autoencoder network was trained with the Poisson noise and Salt-and-Pepper noise one after another after clearing the network parameters.

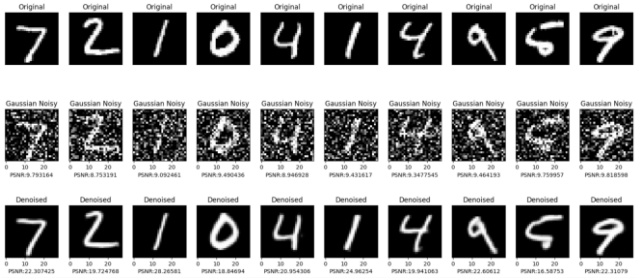


Autoencoder network denoised the speckle noise and reconstructed similar to the ground truth images as shown above. Similarly it reconstructed a near perfect denoised image with the Salt-and-Pepper noise trained network as shown below.

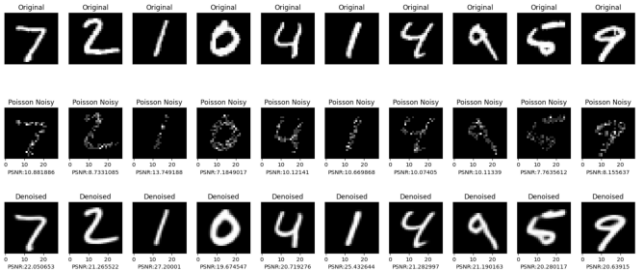


Next the U-Net network is trained in the similar fashion and the results are shown below for each of the considered noise categories.

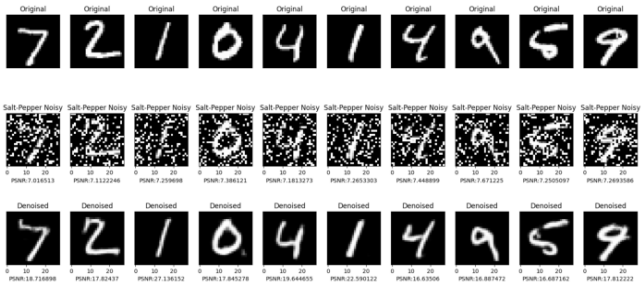
First, results of the denoising Gaussian noisy images with U-Net below



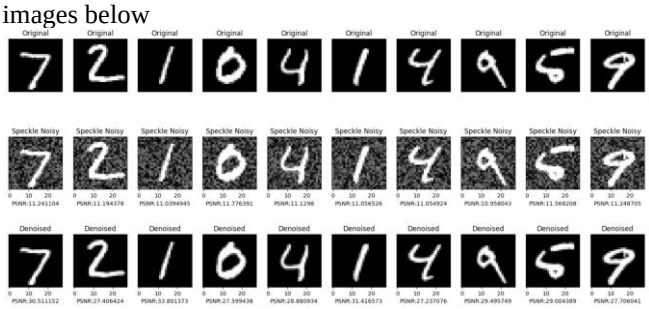
Results of the denoising Poisson noisy images with U-Net below



Results of the denoising Salt-and-Pepper noisy images below with the U-Net architecture



Results of the U-Net architecture denoising Speckle noisy images below

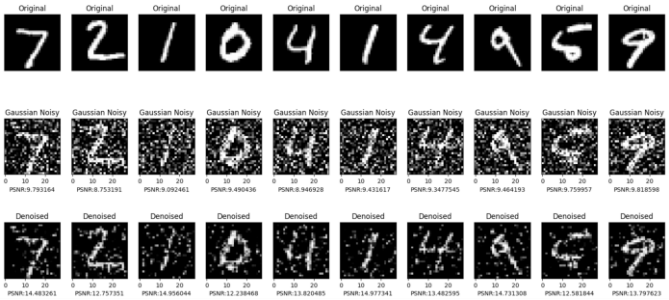


For each of the analyses the PSNR values of all the denoised images were accumulated for both Autoencoder and U-Net networks and the results are shown in the below table.

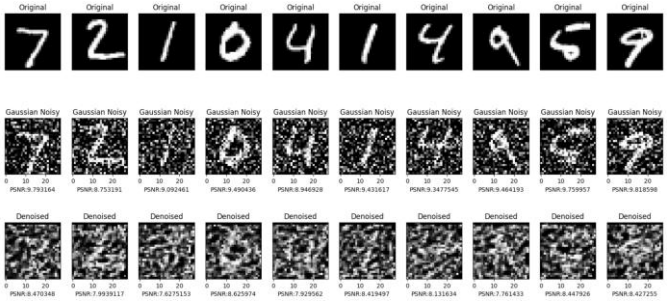
Noise	Autoencoder (PSNR)	U-Net (PSNR)
Gaussian	201686.05	212045.52
Speckle	264605.85	286233.09
Poisson	204246.21	212181.64
Salt-and-Pepper	198724.61	272585.97

From the above table it is clear that the PSNR results are higher for the U-Net architecture for all the noise categories and so we can conclude that the U-Net has performed slightly better than the Autoencoder network. The skip connection that is present in the U-Net network which addresses the limitation of the bottleneck in the latent space representation adds the additional features.

Then I tried denoising Gaussian noisy images on the U-Net model trained with Speckle noise and the results are shown below. In this case the network has not reconstructed well and the output still contains a lot of noise. In this case the network behaves well for the noise it has seen before and fails terribly for the new noise (non-Gaussian).

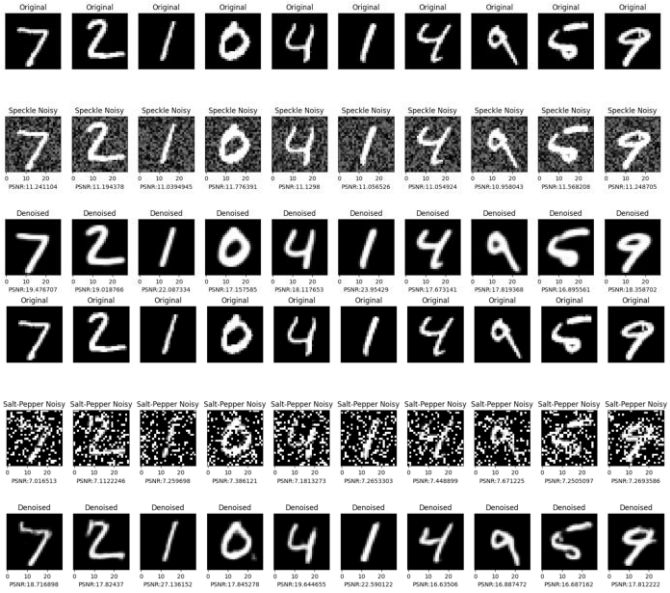


Similarly I tried denoising Gaussian noisy images on the U-Net network trained with Salt-and-Pepper noise and the results are even worse. The reconstructed output in this case contained only noise in this case.

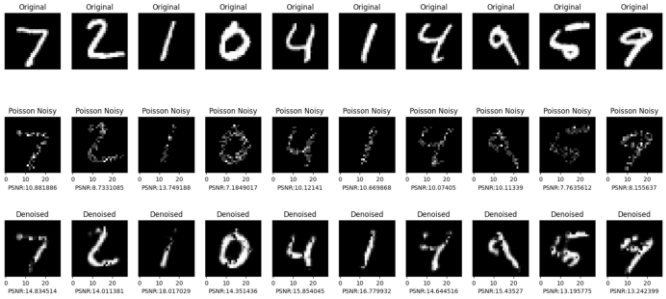


I got similar results when tried with the Autoencoder network for the same configuration.

Finally I tried denoising non-Gaussian noisy images trained on the U-Net network and the results are shown below.



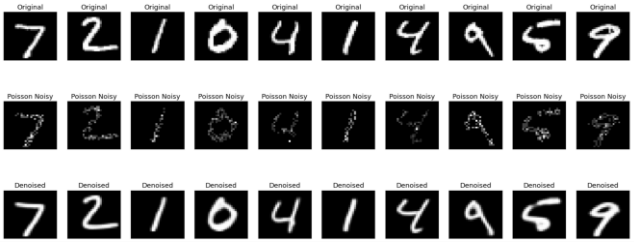
For both Speckle and Salt-and-Pepper noise the U-Net network (and also Autoencoder) denoised non-Gaussian noisy images and reconstructed output close to the ground truth images. However it couldn't denoise the Poisson noise clearly.



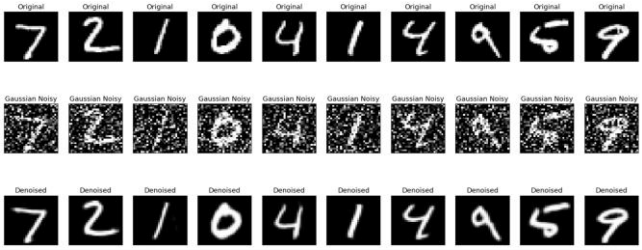
Robust Model

In order to build the robust model I trained the network with all the four noise sources and the results are shown below.

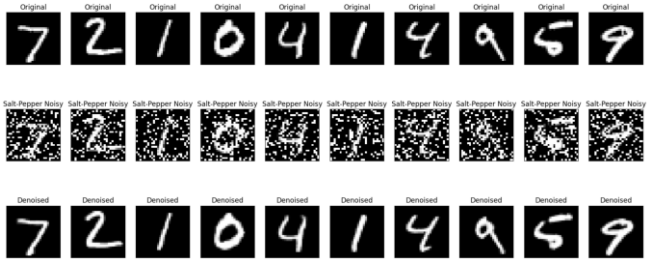
Results of denoising Poisson noisy images with the Robust model below



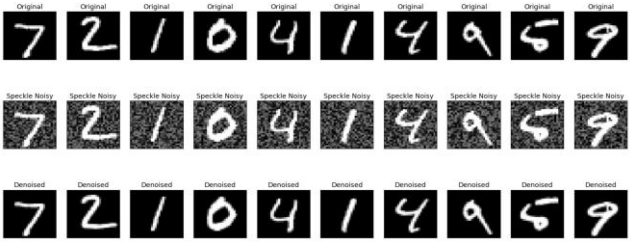
Results of denoising Gaussian noisy images with the Robust model below



Results of denoising Salt-and-Pepper noisy images with the Robust model below



Results of denoising Speckle noisy images with the Robust model below.



I got similar results with both Autoencoder and U–Net when trained with all the four types of noisy images.

4. Conclusion

Based on the PSNR analysis it has been concluded that the U-Net architecture performs slightly better than the Autoencoder network for denoising images. I also found that the networks trained with a particular type of noise performs well only for denoising the same type of noise and for some noises it fails terribly badly. To build a robust model that supports denoising a variety of noise we need to train with most noise sources and such robust models denoise or reconstruct output that looks similar to the ground truth images for all types of noisy images.

References

[1] U-Net: Convolutional Networks for Biomedical Image Segmentation. Olaf Ronneberger et al (2015)