4TN4 Final Project - Noise in Digital Images

Mumuxu Shah
McMaster University
1280 Main St W, Hamilton, ON

Abstract

From Instagram to CT and MRI scans, the demand for digital images to be more clear and accurate has been quite sought after. One of the main explanations of poor image quality comes from digital image noise. Hence, to increase clarity and sharpness in various images, efforts must be made to reduce the noise of a given image all while keeping its originality. There have been a number of denoising techniques on the market, the two most common being the use of filters and kernels or the use of complex deep learning algorithms; each having their advantages and disadvantages. This report implemented support vector machines to try and detect and identify noise in a large grayscale dataset. In our secondary approach we took the same noisy dataset and attempted to remove noise with an AutoEncoder Neural Network. The findings give us a better understanding about the intricacies and complexities of digital noise.

1. Introduction

Digital image noise is an attribute present in all images yet one that is particularly difficult to handle with precision. Most image processing techniques dealing with image noise are either overly simplified or overly complex implementations. The denoising of an image can be done one of two ways; through the implementation of filters and kernels that smooth out the image or by using complex deep learning algorithms. Both of these methods have their disadvantages. Using filters and kernels we are able to reduce unwanted pixel variations, but at the cost of significant image clarity and sharpness, while deep learning algorithms can take hours to train and practically reconstruct your image from scratch. This report will examine the characteristics of noise, determine how it is created, and explore the differences between real and synthetic noise in hopes to get a more thorough understanding of the complexities and difficulties of reducing noise in digital images.

1.1. Noise in Digital Images

”Noise is always present in digital images during image acquisition, coding, transmission, and processing steps”. We can classify noise as the random variation in brightness or color information in an image. Though the definition of noise is simple it can be produced in many ways, and it is very tricky to detect, separate, and eliminate from our images. No image can ever be a hundred percent noise-free.

Noise is generated from every step in the creation of an image starting with a type of noise known as Photon Shot noise. Shot noise is one of the only types of noise where a solution has not been found, as it does not result from any man-made parameters such as camera equipment, or conversion methods. Light-emitting photons have an innate randomness to their movement, they do not travel in a uniform fashion. The best analogy to understand this is to imagine rain falling into a bucket, the water droplets will not hit the bucket/water surface in uniform, they will have their own unpredictable pattern. Similarly, the light will never hit the camera sensor in sync and therefore cannot be recorded as an even number of photons at a given time regardless of the quality of the image sensor itself. This causes variations and unpredictability which is one source of noise. Photon Shot noise is referred to as scene-dependent noise since the incoming speed and volume of photons are dependent on the brightness (the amount of light) in your surroundings.

The next major step of producing an image is to read in information from our camera sensor and then convert that information into the digital intensity values that the computers are familiar with. During this process, the second type of noise known as readout noise can occur. It is scene independent because it relies entirely on our sensors and circuitry. There are two main types of noise in this stage, electronic noise, and quantization noise. Electronic noise is from the pre-analog-to-digital conversion step and has to do with the level of accuracy that a camera sensor can capture information. Counter to that is quantization noise, which plays a role in post-conversion. Quantization noise is the effect of representing an analog continuous signal with a discrete number (digital signal). The rounding error in this conversion is referred to as quantization noise .

Though this covers the major steps in the creation of digital images there are still various other sources for noise to arise. For example, there is something called thermal noise, even with external light entirely cut off from your camera sensor, the heat generated by the camera itself can still create photons and cause noise. This causes many issues during low-light photography and is the reason why all major telescopes are nitrogen cooled. During the processing of images, we encounter a lot of fixed pattern noise, this is because no two pixels can be reproduced perfectly.

Many of the types of noise discussed thus far are what you might interpret as "real noise", it's noise that is sometimes due to nature and often truly random in its occurrence, making it difficult to detect and deal with. However, the models in this report will try to replicate parts of this real noise using added synthetic noise. Since noise is just the variation in pixel brightness or intensity, it can be simulated by changing these pixel intensities ourselves. This study, will review three different kinds of added noise; Gaussian Noise, Salt and Pepper Noise,, and Poisson Noise. If one really looks at it, image noise can be simplified to this one equation:

$$A(x,y) = H(x,y) + B(x,y)$$

Where A is an image with noise, H is a function of noise, and B is a function of the original image. The main objective will be to add and remove H(x,y) to the images as best seen fit.

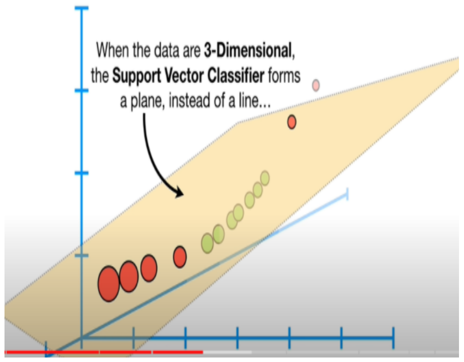
Gaussian noise is a statistical noise, it follows a normal distribution, also known as the Gaussian distribution. Gaussian noise can be used to represent electronic noise as it arises in amplifiers and detectors. The magnitude of Gaussian noise depends on the standard deviation (sigma), the level of noise and the value of sigma which are directly proportional. Another type of noise is salt and pepper noise which is in fact a variation of impulse noise. Salt and pepper noise is the random addition of bright (255 in pixel value) or dark (0-pixel value) intensities added all over animage. This is also known as drop noise. Lastly, we have Poisson noise, the appearance of this noise is seen due to the nature of electromagnetic waves such as x-rays, and visible light. These sources have random fluctuation of photons; hence this can try and replicate the impacts of photon shot noise. All three of these different kinds of noise were added to the imaging dataset.

1.2. Applications of Noise Reduction

The very initial motivations for this report came from the task of creating an algorithm that would be able to accurately estimate the level of noise in medical images. Today image denoising is one of the leading fields of research in the computer vision industry. "Image noise reduction is widely used for various purposes including medical images, industrial non-destructive testing, and remote sensing images" . In the case of medical imaging denoising is particularly important as "doctors diagnose the patient's condition mostly based on the quality of these images" . In order to eliminate noise and continuously achieve reliable results, identifying and isolating noise from the original picture is vital. Unfortunately, doing this is proven to be difficult and there aren't many approaches that can successfully do this. This report will try to use machine learning to first differentiate a noisy image from a clear one, and then take it one step further and identify the type of noise in the image through the use of support vector machines (SVM).

1.3. SVM

SVM is powerful machine learning classifiers, it is a supervised machine learning algorithm and in our first case it will be used as a binary classifier. Using ML is beneficial not only because it is a more smarter and sophisticated approach than any kind of filtering process but it also does not require the time or computational resources demanded by deep learning. Support vector classifiers can be used with lower-dimensional data, likely with no overlap and can work with binary classification. This classifier will find a line of best fit to separate the dataset into two categories. However, when outliers and overlapping datasets need to be accounted for, it is preferred to use support vector machines. SVM classifiers still aim to split your data into two subcategories, though of course, it is not always possible to accurately split datasets. In this case, SVM classifiers use dimensionality to their advantage. In short, the entire mechanics behind SVM is to continuously move the dataset into a higher dimension until it can appropriately split the data using a single hyperplane.


When the data are 3-Dimensional, the Support Vector Classifier forms a plane, instead of a line...

When datasets get more and more complex the issue becomes how does the classifier know which dimension to move the data to? Brute force or trial and error doesn't make for a very efficient method. To solve this, SVM classifiers employ a kernel called the radial basis function. The RBF kernel is a very popular implementation because it can support vector classifications in virtually infinite dimensions. As each new data point is introduced the RBF kernel looks at its nearest neighbors and bases the dimensional classification of the new observation on the classification of its surrounding data points.

Using the SVM classifier we will attempt to detect noise in our images. Afterward we will take the same noisy dataset and attempt to revert it back to its clean state. “The denoising of images corrupted with Gaussian noise is an active research topic in image processing and neural networks. Currently, the best published results come from a deep convolutional architecture that uses hierarchical skip connections” . In our approach, we will be utilizing neural network auto-encoders to denoise our images.

1.4. AutoEncoders

Autoencoders are an unsupervised learning process, the opposite of what we had with our vector state machines. “In its simplest form, an autoencoder is a neural network that attempts to do two things. First, it compresses its input data into a lower dimension, then it tries to use this lower-dimensional representation of the data to recreate the original input” . They capitalize on the fact that structured data that would be seen in our datasets do not fully utilize the dimensional space in which it lies. An image might be represented as a two-dimensional data structure because you can describe its pixels with two (or more) variables, but it doesn’t employ every possible position in that multi-dimensional input space. Autoencoders leverage this and “encode” the input data into its lowest dimensional representation. Then decoders recreate the input data from that lower-dimensional representation given to it by the encoder, their job is to recreate higher dimensional data from a lower-dimensional representation. However, a one-to-one conversion isn’t very meaningful, the point of the middle layer in an autoencoder is to make your data representation even smaller than what is needed to fully represent the input data. Doing this causes information loss which is where all the presumed learning happens. Giving the decoder imperfect data and training the whole network to minimize reconstruction error forces the encoder and decoder to collaborate to find the most effective way to condense data.

Though this is great in theory it can become a bit of an issue as it is difficult for the user to know in advance how much the data can be compressed into a lower dimension. Thankfully, there is a clever adjustment that can be made to get around this issue. Before feeding the encoder the data we add noise to our images, then we ask the network to learn how to erase the noise and reconstruct the original image. Hence, we made sure that the network cannot simply multiply the input by one as this would return the noisy image. In this report, we have tested this approach and will analyze the outcomes of this method of denosing.

2. Proposed Methods

This paper will implement five different models, three of which will be dealing with identifying noise in images and the other two with removing noise from images. A single dataset will be used for all models that comprises 7129 grayscale images of various landscapes.

For the first approach, a SVM classifier will be implemented to perform a binary classification on the dataset. In preparation, three noise generating functions were created, one for Gaussian noise, another for salt and pepper noise, and the last for passion noise. The dataset is prepared in a way that there is an 80% chance that the next image in the dataset will have some form of added noise to it, leaving 20% of the data to remain clean. While adding noise to the prior 80%, each image is feed to a function called $addnoise()$. This function chooses a random scenario from 1-7, each case is associated with a combination of noise that will be added to the image. The cases go from adding just one of the three noises to adding every possibility of mixed noise. For example, “Case 5” refers to the addition of Gaussian plus Poisson noise. Furthermore, each kind of noise is added at a random intensity. For example, the sigma value for Gaussian noise is determined by a random value from 0.02-0.1.

Like this the dataset set of noisy and clear images is created entirely at random. After the noise was added to the dataset, it was then normalized and split using an 85/15 train/test split. In the X train dataset, we placed our randomized images and, in the y train dataset, we have our list of labels consisting of either a 1 or a 0 depending on if the image had noise. These dataset and labels were fed into a SVC classifier and fit the data using an RBF kernel. After the testing was completed, an accuracy score was computed to see if the classifier was successful at predicting whether an image had noise or not.

The second model is much the same in terms of how the data was prepared. The same dataset with the same split and same randomization process was used to add noise. However, there is one distinct difference in this trial, a zero or one was not used to indicate a label for the presence or absence of noise. Instead, each added noise case, from 0-7, was used as the labels. 0 will represent an image with no noise whereas 7 will represent an image with all three kinds of noise mixed. With this trained dataset it can be treated as a multiclass classification problem. Our models can be fitted with the RBF kernel and we can use the one vs rest approach for its classification method. Again, like before we test this and output an accuracy score.

For our last SVM model, we take things back a step. In the last model, we tested our data with all seven noise cases. However, when you mix multiple types of noise together it can become very hard to tell what’s going on in the image. For example, saying if an image had only salt and pepper noise or some mix of all three. In this model, we run all our parameters exactly the same as in our second model but instead, we keep only four cases. Case 0 with no noise, 1 with only Gaussian noise, 2 with only salt and pepper noise, and finally 3 that has an image with only Poisson noise. Lastly, like before after we fit our model, we test it and output our accuracy score.

Next we will attempt implementing a couple deep learning approaches to denoise our dataset. As mentioned before, the same grayscale dataset and technique to randomly add noise to the landscape data will be used. For both the implementations, a convolutional neural network with auto encoder architecture will be employed.

In our first model we import our landscape dataset and parse the data to make sure every image is 150x150 pixels, if any of the images are not this size we remove them as corrupted data. Then like before we create our three functions to add synthetic Gaussian, Salt and Pepper, and Poisson noise. For our first denoising model we will not implement any mixed noise, as to keep the task simple, and like before we will only be working in grayscale. When creating our encoders and decoders we keep three convolutional layers in our network. In our approach we will use 3 by 3 kernel size and relu activation for all layers. In encoder we reshape our image down to a 19 by 19 format and the reverse the shape back to 150 by 150 during the decoding process

Once our encoders and decoders have been created, noise is added to the entirety of our dataset and then our dataset is modified so it can fit our network. To accomplish this our data is first split into an 80% training portion and a 20% testing portion. We then reshape it to fit the shape (Num Of Images, 150, 150, 1) and then we turn that matrix into float type value. At this point, the data is ready to be fitted and trained over our neural network for 100 epochs. The last step is to use our trained network to predict our images without any noise and display them to see our results.

For the final model the code is executed in a very similar fashion with only one major difference. This time the data is trained over the same network with the same parameters, but only white gaussian noise was added to our images. The data was simplified to only one kind of noise and see how the networks perform.

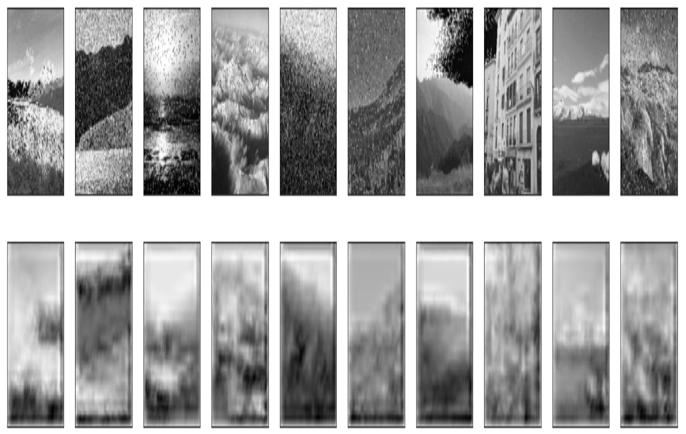
2.1. Experimental Results

For our first model of implementing SVM with a binary classification we got an accuracy score of 0.794392523364486. This tells us that when it comes to identifying whether an image is noisy or clear our model can predict this with a reliable accuracy or nearly 80%.

Our second model was implementing an SVM multiclass classifier over the full range of added noise cases. The Goal of this classifier is to match the noisy image with which of the eight noise cases it was assigned. For example, if it was an image with only salt and pepper noise it should be able to predict that it belongs to class 2. After training this model we received an accuracy score of 0.21401869158878506. From this we can see that this was simply far too complex of a task for the classifier to achieve and that is shown in our low 21% accuracy score. This was to be expected however because when we start mixing different noises into an image it is difficult for even us to tell them apart and exactly identify their noise composition.

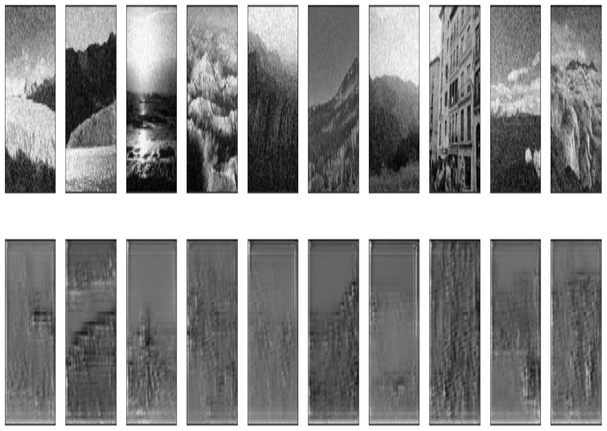
Our third model is our final machine learning model implementing SVM. In this model we continue to use our SVM classifier to multiclass classifications, this classifier has the same parameters as the previous with an RBF kernel and a one vs rest implementation. However due to the previous results we decided to make our classification much simpler and brought it down to only 4 classes. We now want the predictor to be able to guess if the image is with added Gaussian noise, salt and pepper noise, Poisson noise or without any noise. From this model we received an accuracy score of 0.28785046728971964. From making our classification simpler we expected to get a better score and we did get about an 7% increase in our accuracy, but this is still too low to be considered a successful model. Turns out even with simplified added noises it is still too difficult for the computer to detect the kind of noise we use. I feel this may be in part due to our complex dataset that we use.

The fourth implementation was the first of our deep learning models and it had an entirely different task to perform. Here our object was not to detect noise but to remove it and we implemented this with our 3-layer auto encoders. Here are side by side results of our image with noise and the predicted image from our network.



As we can see our images are very blurry, almost unrecognizable.

Therefore, for our 5th model we simplify the dataset greatly. In our last model we had already taken out mixed noise from the equation but in this test, we will implement only added white Gaussian noise to our images and nothing else. Here are the results we see this time, displayed just like before.



Once again, we can see that our image is really poor, almost worse than before. One major reason for this would be the structure of our neural network, when decreasing our image size using our encoder, we only shrink the image down to 19x19. Whereas we should have kept a few extra layers to our network. Additionally, this entire network might be a little simplified for the complex dataset we were using causing it to not give us reliable or accurate results. It is important to note that all other implementations similar to this are implementing this network onto the minst dataset which is a considerably easier dataset to work upon.

2.2. Conclusion

Through this research report and our several implementations, a thorough understanding of the details and complexities of noise in digital images was achieved. We learned about the difference between read and synthetic noise and how we can attempt to mimic real noise in our code. After understanding the foundations of support vector machines, we used this classification method to detect noise in our dataset. From doing this we noticed that though it is fairly easy to identify if an image is noisy yet, it is extremely difficult to pinpoint and label different types of noise. Currently the noise we are working with is a type of added synthetic noise, simply changing pixel values in our image matrix. However, as we know real noise is far more elusive than that and these short examples can give us more of an appreciation towards the complexities of being able to detect noise. It can show us why it is such a big field of research in the industry. Next in our research we attempted to understand the correlations of deep learning in regard to computer vision. We studied convolutional neural networks, more specifically auto encoders and learned how they can be used to denoise an image. We used our knowledge of auto encoders to create a simple CNN model to denoise our landscape dataset. Both of our attempts towards this failed unfortunately which brought us to the realization

Method	Frobnability
Theirs	Frumpy
Yours	Frobbly
Ours	Makes one's heart Frob

Table 1. Results. Ours is better.

that our network was overly simplified for the complicated task we gave it to handle. Though our classification parameters were adequate our shortcoming came in the number of convolutional layers we kept in our network. When we are shrinking and reconstructing an image, dimensionality plays a critical role. By how and how often we reduce the dimensions of our image in our encoder makes or breaks our final product. The sophistication of our network we can safely assume would be enough to reconstruct a simple binary image such as digits from the MNIST dataset. In the end our first model was still entirely successful and could be used as a preprocessing step in a larger algorithm. Many times, we have very large datasets and tasks that one needs to work on, it is important to sweep through your data to see if any images are noisy so you can either filter or discard them. Moving forward we would like to improve upon these ML models that we made so we can bring ourselves closer to eventually being able to create a model that could effectively detect and estimate the noise level of an image. We would also like to see how these models fare on different datasets, see how they perform on colored images or medical images. In the end this was an incredibly useful research project to help introduce us to the world of digital noise, and hopefully, enough to motivate us to continue our industrial quest to create a near perfect image with little to no noise*.

*so far because of problems like photon shot noise we believe no image can be perfectly clear, but I say we're capable of more than we know and nothing is impossible.

References

[1] Yuqian Zhou, Jianbo Jiao, Haibin Huang, Yang Wang, Jue Wang, Honghui Shi, Thomas Huang, "When AWGN-based Denoiser Meets Real Noises" Available: https://jianbojiao.com/pdfs/AAAI.pdf.

[2] Baozhong LIU, Jianbin LIU, "Overview of image noise reduction based on non-local mean algorithm"

[3] Lev Yasenko, Yaroslav Klyatchenko, Oksana Tarasenko-Klyatchenko, "Image noise reduction by denoising autoencoder" Available: