Uncovering the Signal: A Comparison of Noise Reduction Methods in Digital Image Processing

Chelsea Everest
Faculty of Engineering
University of Western Ontario
London, Canada
ceveres4@uwo.ca

Alexander Guoth
Faculty of Science
University of Western Ontario
London, Canada
aguoth@uwo.ca

Rachel Vanderloop
Faculty of Engineering
University of Western Ontario
London, Canada
rvanderl@uwo.ca

***Abstract*—The denoising of images is a well-known and studied problem. Applications for denoising can be found in nearly every field, and each new problem needs to be considered individually, as every imaging system deals with unique and varying noise sources. In this study, we examined three common ways of removing different types of noise from images: Gaussian convolution of linear smoothing filters (Gaussian smoothing), principal component analysis (PCA), and an autoencoder neural network. Though there is a lot of research done on these algorithms individually, there is a lack of research comparing them directly against different types of noise. To compare these three methods, we created two noise added datasets and measured the Mean Squared Error (MSE) of the dataset reconstructions. For Gaussian noise, Gaussian smoothing produced the least visually appealing results but produced the lowest reconstruction error. For salt and pepper noise, PCA produced the lowest reconstruction error. Though it was judged that the autoencoder was the best performer based on visual results. In future work, a larger dataset should be used with more powerful hardware, so the reconstruction potential of our learned methods (PCA and autoencoder) is maximized.**

***Keywords*—PCA, Autoencoder, Gaussian Smoothing, Denoising, Salt and Pepper Noise, Gaussian Noise**

I. Introduction

Image denoising is a well-known problem that has been heavily studied as it plays an important role to many applications from image reconstruction to image classification and computer vision [1]. Image denoising is the process of reconstructing an original image, or a more clear version of an image, from a noise corrupted version. Noise can be caused by many different sources, but in nearly every field there are specific denoising processes based on the unique standards for the final image. Every situation needs to be considered individually given the wide array of imaging technologies in use and the many physical processes that introduce noise.

Noise can come in a variety of forms, and not all of them will be significant to every problem. As such, a diverse range of denoising algorithms may be required for a specific application. Denoising is an important aspect of preprocessing in many different computer vision problems, like Gaussian smoothing prior to edge detection. There is a significant amount of research that has been done in the area of denoising images with the methods we will discuss, but little that compares them directly against different types of image noise. Our goal will be to compare the reconstruction performance of each denoising method on datasets with different types of simulated noise added to them, in order to determine which denoising algorithm is best suited for which noise specific application.

II. Related Work

Image noise remains a significant issue for many different applications of machine learning and computer vision algorithms. Research is active, and new algorithms are being created in an attempt to manage this problem. There are many different noise reduction algorithms, and each has their own specific advantages and disadvantages depending on the application and requirements. Some methods include non-local means, wavelet transforms, and block-matching algorithms. Here, we will be discussing PCA, autoencoders, and Gaussian smoothing. These algorithms are used in a variety of applications including Canny edge detection, and general noise reduction, to name a few.

Related work in the field of PCA includes [2] where a novel method is considered for denoising photon-limited images using an adapted version of PCA. Dictionary learning and sparse patch-based representations of images are used to denoise these photon-limited images. This research is important for images taken in low-light situations, including spectral imaging and nuclear medicine.

Another recent study was done on removing watermarks from images [3]. This research is similar to denoising, but the “noise” being eliminated is an applied watermark. They decompose the image into multiple levels and process it to remove the watermark and restore the original image. The new algorithm that they propose shows promise against signal processing attacks and improves upon existing algorithms that are used for the same purpose.

Additionally, a study published earlier this year [4] discussed how the noise in cryo-electron microscopy images can be removed using a cascade of denoising autoencoders.

The noise from these images is complex and comes from many different sources, making it very challenging to remove. This noise is removed in a progressive manner using separate blocks of neural networks that contain a convolutional encoder. Images like these are used in structural biology, and are important for the study of cellular structures.

Finally, the last denoising method that we tested was Gaussian smoothing convolution. This method is often used for reducing the noise in an image by convolving it with a Gaussian kernel, and is an essential component of the Canny edge detector. Canny edge detection is an important algorithm for identifying structures in images, and is ubiquitous in the field of digital image processing. For example, in [5] it is combined with YOLOv3 Object Detection to detect accidents on the highway and analyse the severity of any accidents it detects.

III. Methods

The main objectives of our study are as follows:

1) Investigate different types of noise that affect digital images.
2) Evaluate a variety of common denoising algorithms, including passive and learned approaches, on different types of image noise.
3) Determine which methods perform best on which types of noise and understand why.

It was important to use noise that is most often encountered in real world imaging situations such as a picture taken in low-light conditions or a scanner with physical flaws. For this reason, we decided that the two types of noise we would experiment with would be Gaussian and salt and pepper, which are both pervasive in an imaging context [6]. Our chosen noise reduction methods are varied, representing both passive and learned techniques, with the Gaussian convolution being the passively applied process and PCA and Autoencoder requiring learning before application. Our learned techniques are also varied, with PCA being an unsupervised method and the autoencoder requiring supervised training. For our error metric, we measure the MSE for each of the reconstructed datasets produced by our methods.

A. *Data*

The dataset that we utilized was the MNIST dataset [7], which is a database of 70,000 handwritten digits; 60,000 training images and a test set of 10,000 images. The dataset contains ten image classes representing the digits 0 to 9. The images in the dataset have all been centered, scaled to a size of 28x28 pixels, converted to greyscale, and normalized to improve training and error metric evaluation. The images from this database are ubiquitous in machine learning given the size, quality and availability of the dataset which is also the reason we chose it for our study.

Per the requirements of our study, noise needed to be added to the original images from the dataset to evaluate the noise reduction performance of each method. Prior to adding noise, we first normalized the pixel values in each image to standardize our comparisons. To add Gaussian noise, we used NumPy [8] to sample vectors from a Gaussian normal distribution with a mean of 0 and a standard deviation of 0.25. These vectors were then added to images, with any resulting pixels having a value >1 clipped. For salt and pepper noise, we again used NumPy [8] to sample two vectors from a Gaussian normal distribution with a mean of 0 and a standard deviation of 0.25. With the two random vectors, we kept only the values greater than 1.5 standard deviations, setting those values to 1 in the first vector and 0 in the second, with the remaining pixels having the opposite value. The result is two vectors containing randomly distributed black and white pixels, respectively. The vector containing black pixels was then added to the image which was then multiplied by the vector containing white pixels, with any resulting pixels having a value >1 clipped. This produced images with randomly distributed black and white pixels, meeting the criteria for salt and pepper noise.

B. *Models*

Gaussian smoothing is the standard passive approach to noise reduction in digital image processing, which is the reason we chose it as one of our methods. For our study we used the convolution functions available in SciPy [9] and decided on a kernel with a standard deviation of 0.25, which was suitable for the smaller image sizes from the dataset. The reconstructions for this method were produced by applying Gaussian smoothing to both the gaussian and salt and pepper noise added datasets.

In practice, PCA is often used for reducing the dimensionality of data, which can then be reconstructed to full dimensionality using the principal components. However, the characteristics that make PCA effective at dimensionality reduction also make PCA effective as a noise reduction technique. Using the principal components produced by PCA, we can reconstruct noised images by the same process used to reconstruct dimensionally reduced data. We chose this method due to its ubiquity and its unsupervised approach to learning. Using the PCA implementation in SKlearn [10], we extracted the first 500 principal components from 10 000 images from each of the noise added datasets. Choosing the first 500 principal components was a good compromise between performance and generalizability. The reconstructions for this method were produced by applying PCA reconstruction with the principal components to each image in their respective noise added dataset.

Our last noise reduction method, the autoencoder, is commonly used for data denoising [11] and ReLU activation is a standard hidden layer for neural networks [12]. The model we used for our autoencoder has an input layer size of 784 (28x28) as it is the size of our input images. The linear input layer reshapes our data from 784 to 128 so that it is smaller than the input layer as this data compression makes it much easier for the network to extract only the most important information which, for our case, is the general shape of the digits while ignoring the noise. Following the linear activation, the first hidden layer uses the rectified linear unit (ReLU) activation function to determine the relative importance of the input features. ReLU is a good option as a hidden layer for our neural network as no units will be negative which avoids the common vanishing gradient problem [13] that is seen with other activation functions such as Sigmoid and tanh. ReLU also has a very simple gradient which is a benefit as we experienced issues with hardware limitations. The last hidden layer consists of a linear unit to increase our ReLU output back up to the size of our desired images from 128 back to 784. Moving to the output layer, we chose Sigmoid as this activation function outputs a value between 0 and 1 which is also the range that our pixels have for colour values. With 0 being white and 1 being black, the autoencoder having an output function as Sigmoid works perfectly for our normalized dataset. Finally, it was decided that we would use a fully connected neural network over a locally connected neural network that is commonly used for image classification. Since our model is quite simple, computational cost won't be as high as a neural network with multiple hidden layers. Another factor is our dataset; since it is normalized and only grayscale images this also decreases general computational costs making a fully connected neural network a viable option. With the way that neural networks learn, fully connected layers can become locally connected if the weights of the units are learned to be 0 but choosing a fully connected model meant that the assumption that only local pixels are crucial to the denoising process did not have to be made and the weights could be adjusted appropriately throughout the learning process.

C. *Evaluation*

To measure the performance of our methods, we used MSE which measures the average squared difference between the estimated values and the actual value. The formal definition is given below.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \qquad (1)$$

Where n is the total number of samples in the dataset, Y is the expected value and $\hat{Y}$ is the predicted value. For our purpose, MSE was a good metric as it quantifies the pixel-to-pixel differences between the original image and the denoised image returned by our models. These pixel differences are then squared and averaged which provides a good general approximation of reconstruction error, for our purposes. Additionally, MSE is a broadly used error metric for comparing image vectors and is generally considered robust and effective.

IV. Results

To evaluate how well our three noise reduction methods apply denoising against our two types of noise we established three basic comparisons. First, to have a visual intuition for how the different reconstruction methods affect the original data, we applied t-SNE [14] dimensionality reduction to each of our datasets, including original, noise-added and reconstructed. Next, to quantify how each reconstruction method performs, we evaluated the Mean Squared Error (MSE) of the noise-added (as a baseline comparison) and reconstructed datasets against our original training dataset. Finally, to quantify the generalizability of our two learned models, that is PCA and autoencoder trained on our training set, we again evaluated their MSE, but this time against our test dataset.

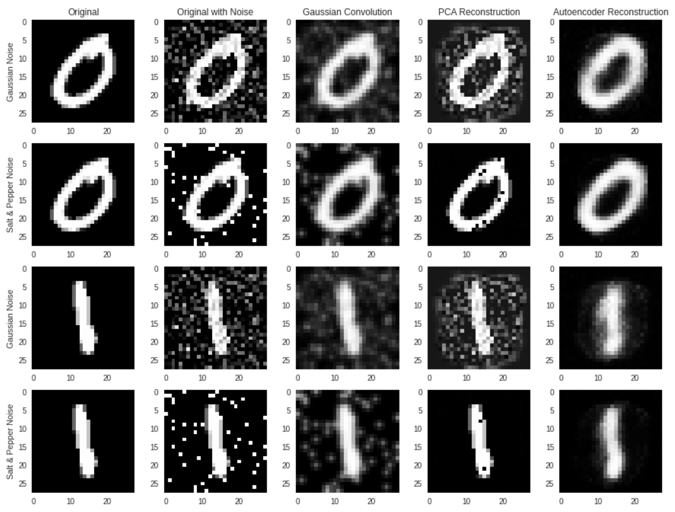

Fig. 1. Reconstruction models with original images

A. *At a glance*

If we examine the outputs of our three reconstruction methods based solely on their visual merit, it is clear to see that the autoencoder, overall, produced the most readable results, especially for the salt and pepper noise samples, as shown in Figure 1. Overall, salt and pepper seemed to be the easiest type of noise for our learned models to correct for, with PCA also producing highly readable reconstructions of the original image. All models seemed to be most challenged by the Gaussian noise samples. The autoencoder showed fairly inconsistent behaviour with these samples, with some reconstructions being of very high quality and others being nearly illegible; the overall effect is probably best described as a 'smearing'. Conversely, the PCA reconstructions were consistent, but each includes a patch of noise in the immediate radial area around the number, producing an 'aura-like' effect that greatly affects the legibility in some samples. For both types of noise, the samples convolved with the Gaussian filter had a very consistent and predictable effect, best described as a 'softening' of the original, noise-added image, producing consistently readable reconstructions. A curiosity to note is the occasional dark pixel added in the PCA reconstructions of the images with salt and pepper noise added. This artifact is consistent across all reconstructions but has little to no effect on its readability.

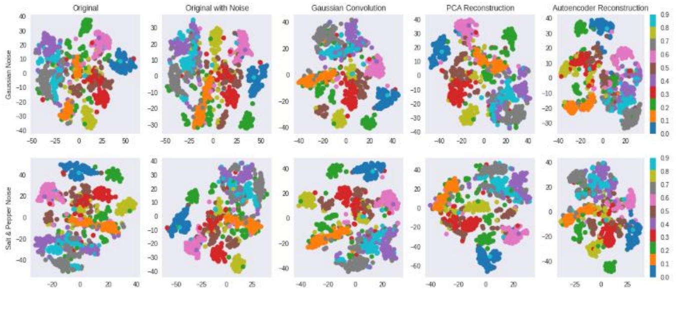

Fig. 2. t-SNE reductions of our initial and reconstructed datasets

B. *Visualizing the effects*

To better understand the effect that each reconstruction method has, we plotted the t-SNE reductions of our initial and reconstructed datasets which are shown in Figure 2. These plots do not have a fixed perspective which can make them challenging to interpret, but we can select a specific case and observe how it changes after each transformation. For the Gaussian noise, we can see that the images of fives (coloured brown in the plots) are the written numbers most affected by the application of the noise, becoming significantly more diffuse in the noise added plot than in the original. With the convolved reconstructions, the 'fives' are noticeably more clustered, but still distant. Conversely, both the PCA and autoencoder reconstructions produce much more tightly clustered 'fives' that more closely resemble the clustering behaviour seen in the original data. This is consistent with the visual results shown in Figure 1. For the salt and pepper noise, we can see that the images of threes (colored red in the plots) are the written numbers most affected by the application of the noise, with the two distinct clusters shown the original becoming more separated after the noise is added. All three reconstructions appear to re-cluster the 'threes', with the PCA reconstructions producing a cluster that most closely resembles the cluster shown in the original plot. However, both the convolution and autoencoder reconstructions also produce clusters very similar to what is seen in the original. Again, this is consistent with the results shown in Figure 1.

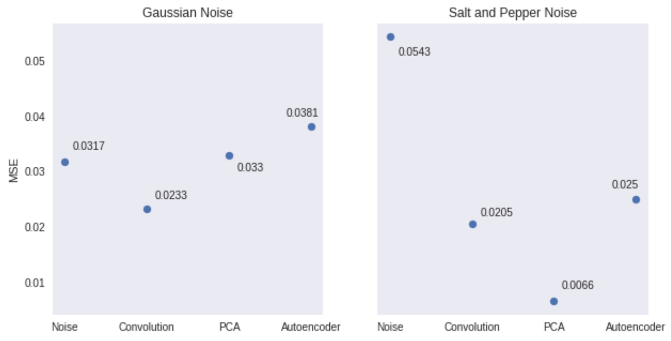

Fig. 3. MSE of each reconstruction with the original training set data, along with the noise-added samples

C. *Understanding the effects*

To quantify the performance of each reconstruction method, we evaluated the MSE of each reconstruction with the original training set data, along with the noise-added samples to serve as a baseline for comparison. These results are shown in Figure 3. Interestingly, for the data to which Gaussian noise was added, the method producing the least visually appealing results, convolution, actually produced the lowest reconstruction error, followed by PCA then autoencoder. More interestingly, the reconstruction errors of PCA and autoencoder, the methods producing the most visually appealing reconstructions, were worse than the noise-only error, albeit marginally. For the data to which salt and pepper noise was added, the results are much more consistent with what we see. Here, PCA produced the lowest reconstruction error followed by convolution, then autoencoder, with each showing improvements over the baseline. Similar to the Gaussian noise case, the autoencoder performed marginally worse than the other three methods, even though we judged it to be the best performer based on its visual results.

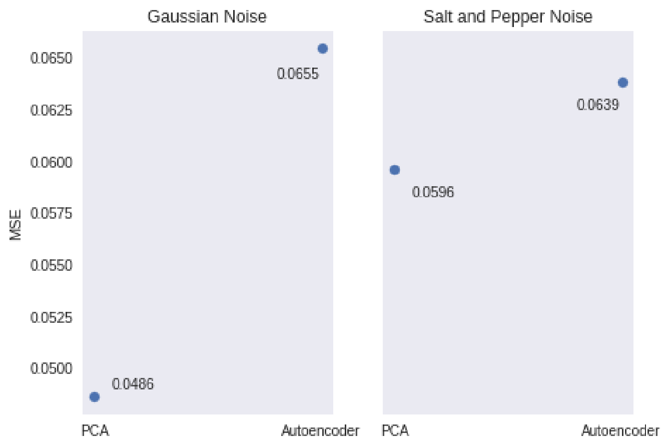

Fig. 4. MSE on our noise-added test dataset for PCA and autoencoder

D. *Generalizing the results*

To put our results in context, we compared the performance of both our learned reconstruction methods, PCA and autoencoder, using MSE on our noise-added test dataset. The results are shown in Figure 4. For both the data with added Gaussian and salt and pepper noise, PCA had a better reconstruction error than the autoencoder. Interestingly, the difference between errors of the two methods was opposite of what was observed in the training data, with PCA significantly outperforming the autoencoder on the Gaussian noise and only marginally outperforming it on the salt and pepper noise. This supports what we have seen in the previous results which show the autoencoder struggling more than the other methods when reconstructing the Gaussian noise-added images. As expected, both models performed worse on the test data, but not significantly.

V. Conclusion

In conclusion, it was interesting to compare the algorithms and see how well they do against each other for different types of noise. Gaussian smoothing had the best results for Gaussian noise, and PCA had the best results for salt and pepper noise, but visually the autoencoder seemed to perform best. Of the two types of noise that were used, the salt and pepper noise was the easiest to manage. We believe this is because a lower fraction of the total pixels are affected by the addition of noise and because the values of corrupted pixels have little variation, being either completely black (1) or completely white (0). For the learned methods, this makes noise corrupted pixels much easier to recognize. For all methods, Gaussian noise was more difficult to manage, which we assume is because nearly all pixels are affected to a certain degree. Since the degree and distribution of corrupted pixels varies so much, the learned methods have much more difficulty recognizing the corrupted pixels. For this reason, simple Gaussian smoothing performed best by just redistributing the noise over all pixels.

A. *Lessons Learned*

The hardware that was used in our study was certainly a limiting factor. Only 20,000 of the original 60,000 images were used to train our learned models, as that was the limit our implementations were capable of handling with the resources that were allocated to it. Because of this, the reconstruction potential of our learned models was not maximized which is likely a contributing factor to the unexpected performance on the Gaussian noise added dataset.

B. *Future work*

There are several different ways that the results could be improved in the future. As stated previously, only a subset of the available images were used in training the models, as a result of hardware limitations. If we had the processing power to use the entire dataset then, for PCA, the principal components that were generated could have captured more variance. Additionally, the autoencoder would have certainly seen performance benefits if more training data had been used. This, as with the previous problem, could be solved with an upgrade in hardware in addition to low-level code optimizations. In future research, we hope to replicate our study on a larger dataset with improved hardware capable of processing it. It would also be interesting to experiment with new types of noise to have a more complete understanding of which methods are best suited for which applications.

References

[1] F. Röhrbein, P. Goddard, M. Schneider, G. James and K. Guo, "How does image noise affect actual and predicted human gaze allocation in assessing image quality?," *Vision Research,* vol. 112, no. ISSN 0042-6989, pp. 11-25, 2015.

[2] J. Salmon, Z. Harmany, C.-A. Deledalle and R. Willett, "Poisson Noise Reduction with Non-local PCA," *Journal of Mathematical Imaging and Vision,* vol. 48, no. 279-294, 2014.

[3] V. S. Verma, A. Bhardwaj and R. K. Jha, "A new scheme for watermark extraction using combined noise-induced resonance and support vector machine with PCA based feature reduction," *Multimedia Tools and Applications,* vol. 78, pp. 23203-23224, 2019.

[4] H. Lei and Y. Yang, "CDAE: A Cascade of Denoising Autoencoders for Noise Reduction in the Clustering of Single-Particle Cryo-EM Images," *Frontiers in Genetics,* vol. 11, 2020.

[5] Y.-L. Chung and C.-K. Lin, "Application of a Model that Combines the YOLOv3 Object Detection Algorithm and Canny Edge Detection Algorithm to Detect Highway Accidents.," *Symmetry,* vol. 1875, no. 11, 2020.

[6] A. K. Boyat and B. K. Joshi, "A REVIEW PAPER: NOISE MODELS IN DIGITAL IMAGE PROCESSING," *Signal & Image Processing : An International Journal (SIPIJ),* vol. 6, no. 2, 2015.

[7] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/.

[8] C. R. Harris, K. J. Millman and S. J. van der Walt, "Array programming with NumPy," *Nature,* vol. 585, no. 7825, pp. 357-362, 2020.

[9] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson and W. Weckesser, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods,* vol. 17, pp. 261-272, 2020.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Tirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.

[11] Q. Ma, W.-C. Lee, T.-Y. Fu, Y. Gu and G. Yu, "MIDIA: exploring denoising autoencoders for missing data imputation," *Data Mining and Knowledge Discovery,* vol. 34, no. 6, p. 1859, 2020.

[12] Z. Chen and P.-H. Ho, "Global-connected network with generalized ReLU activation," *Pattern Recognition,* vol. 96, no. ISSN 0031-3203, 2019.

[13] Z. Hu, J. Zhang and Y. Ge, "Handling Vanishing Gradient Problem Usising Artificial Derivative," *IEEE Access,* vol. 9, pp. 22371-22377, 2021.

[14] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research,* vol. 9, pp. 2579-2605, 2008.