Jaypee Institute of Information Technology, Noida



MINOR PROJECT   
(Synopsis)   
  
Privacy Preserving Data Mining

Team Members:   
Dharmesh Malav - 19803005  
Nikhil Paleti - 19803024  
Divyanshu Tiwari – 19803025

OBJECTIVE

The objective of the proposed project is to look into and investigate about Privacy Preserving Techniques that are existing, and employed in the real world. The proposed project would also evaluate the privacy preserving techniques as deployed, on a real life dataset.

INTRODUCTION

Privacy is the point of conversation of many people lately, because of the complexity, scale and importance of data, more specifically – Personal Data, in the current era of computing, which relies heavily on such data to train Soft Computing models and personalize our experiences.

This data, however, is often stored as plain text, often unsecured, on databases, to be processed as quickly as required.  
This calls for research and work to be done on techniques to protect the privacy of the users who have their data spread on the internet.

There are 3 major privacy preserving techniques that have been coined and researched to an extent.

1. K-anonymity:
2. L-diversity:
3. T-closeness:

However, these are still completely unused in the real world, on the field of data mining, even by major organizations like Apple, Microsoft or Facebook, who collect user data on a large scale

Experimental Design

The process of the project can be broken down into the following steps:

1. Data Pre-processing
   1. Ingesting CSV files
   2. Convert CSV to Graph or np.array (As required)
2. Apply HITS/PageRank algorithm on the dataset to acquire scores.
3. Apply Privacy Preserving Techniques (k-Anonymity) on the dataset.
4. Re-Apply HITS/Page-Rank to get scores again.
5. Compare scores from first and second HITS/PageRank and trace-ability to evaluate applied privacy metrics.

Datasets and their Analysis

1. Co-Authorship Network Analysis Dataset  
   <https://www.kaggle.com/code/bkoseoglu/co-authorship-network-analysis/data>

Research Variables

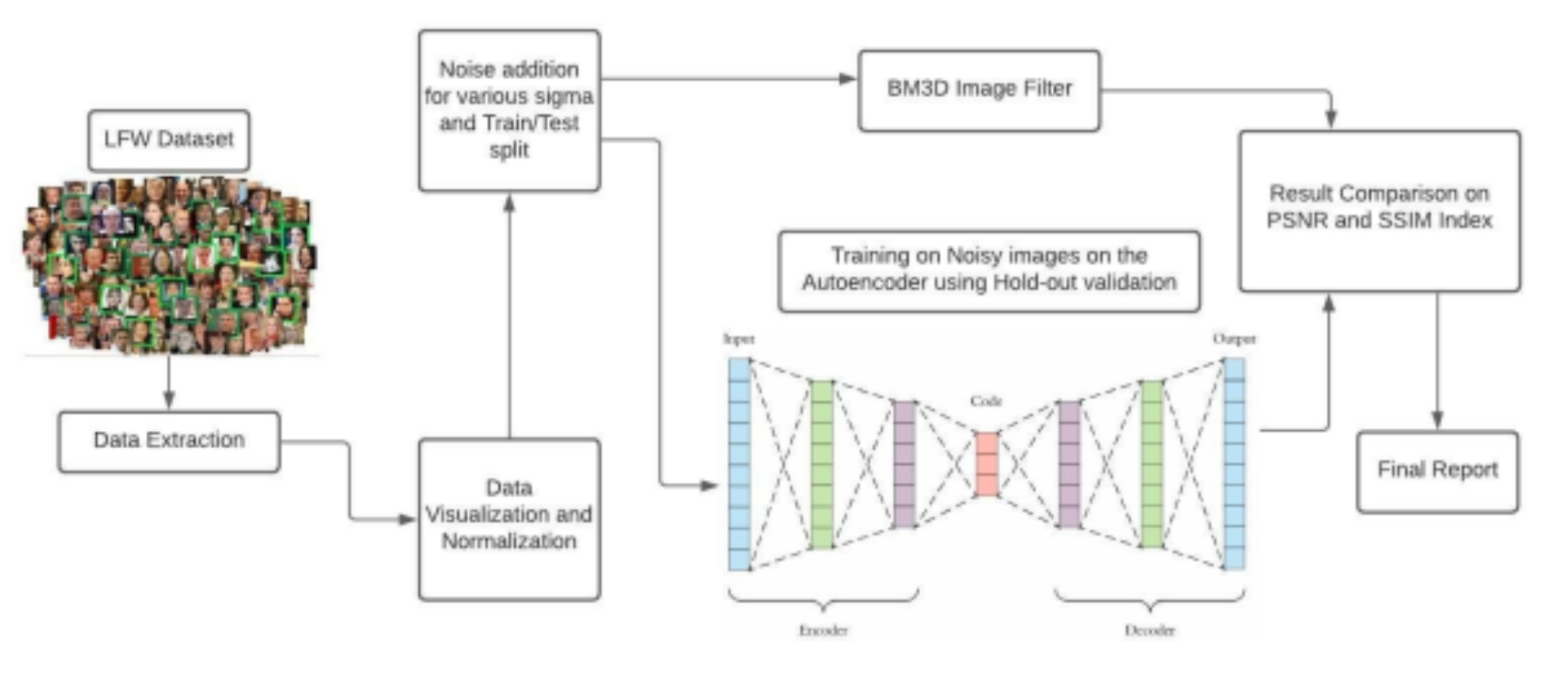
Auto-Encoders are unsupervised learning techniques in which we leverage neural networks for the task of representation learning.

The independent variable for this model is the input image that has been infused with added Gaussian noise, and the dependent variable is the resulting de-noised image.

We have used 2 image sizes, 32x32 and 64x64 for our final Model. Another important variable is the noise factor, we will use 2 factors - 0.1 and 0.2. The final variable for the model is the kernel size, we have used and experimented on 2 variants, (3, 3) kernel size and (5, 5) kernel size, what we mean by it is the size of the kernel used in CNN layers of our model, the bigger the size the more image is covered for its convolution operations.

The other variable is the Latent variable, the central representation of the data that includes its most important features just like PCA, its dependent on the input as well.

Project Procedure Diagram



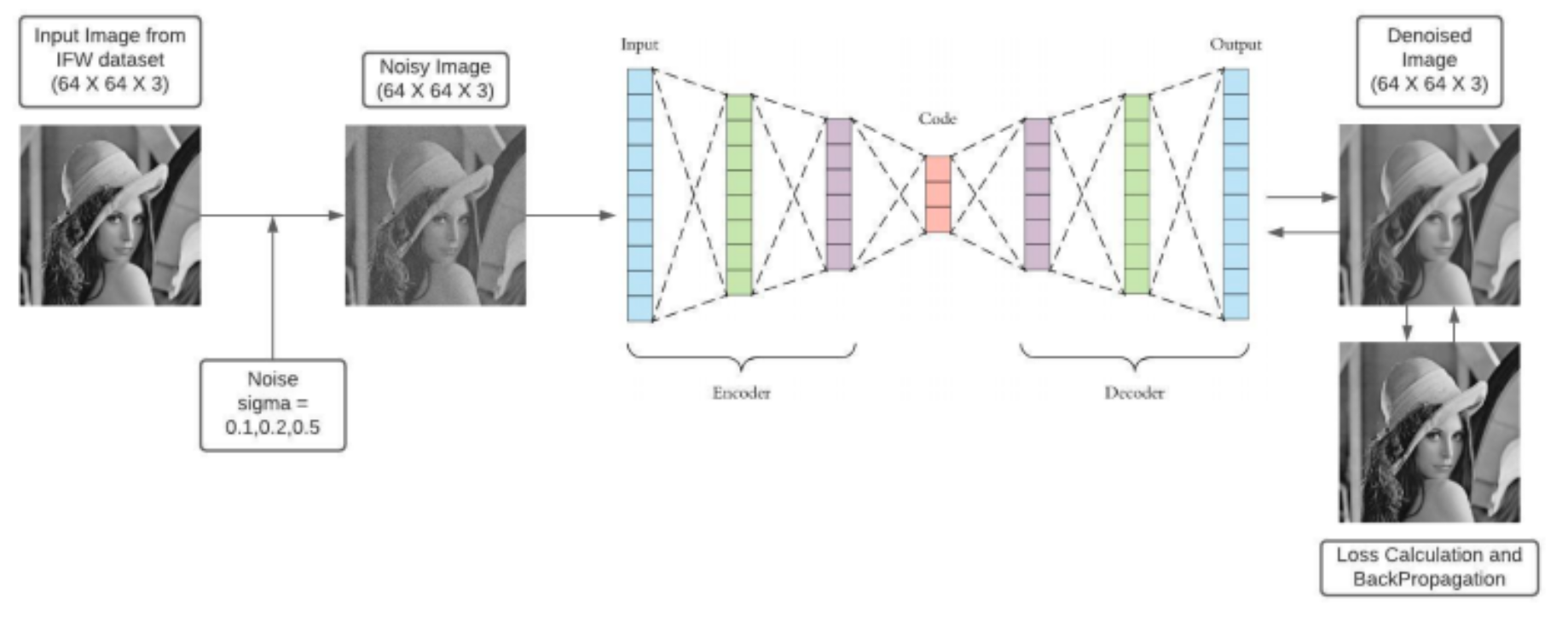
As the Project flow suggests, the first process executed for each of our models and experiments is the data preparation, adding noise to it creating ground truth images to be compared later to the outputs.

Then the data is divided and used in training and testing models and the results are then compared to that of BM3D.

For the complete test dataset, the images are first run through our model and then they are run through the BM3D function that generates the output image as well.

Both these images are then passed through the PSNR and SSIM functions are used to generate scores for each of these images, and finally a mean score for that particular dataset is generated and recorded.

Model Diagram



The Model we use is a CNN-powered Auto-Encoder, it uses 3 CNN layers in encoder for 64x64 images and 4 for 32x32 images, same for the decoder, which uses stacked Transpose layers.

The architecture is such that the middle ‘encoding’ has the best chance to capture meaningful features from the images, and use them to recreate the images without noise, for 64x64 its size is 128 as the features are more abundant, for 32x32 it is 32.

The loss functions used and optimizers deployed are discussed later.

Data Analysis

1. The images are obtained from the Kaggle in the form of zip, txt and csv formats.   
2. The images are extracted from the zip files and their corresponding labels as well.   
3. We don’t need the attributes csv file for this project.  
4. The data is firstly visualized for outliers and the standard deviation values are fixed to crop the dataset for outliers.

Research Methodology

Data Preprocessing Techniques

1. The first preprocessing technique for any image dataset is its visualization, we have used tensorflow for plotting the RGB scatter plot to detect outliers.

2. All the images below or above the threshold deviation values are removed.

3. The images sizes are fixed and stored in variables.

4. The data is normalized.

5. The Training data is formed by adding noise at various sigma values to original images.

6. The images are then converted from BGR to RGB.

7. Normalization of images is done from

8. Now, for the training data, we introduce Gaussian Noise at different rates \_\_\_\_VALUES\_\_\_\_

9. The final dataset is then split into train and test for model training.

Before Pre-processing After Pre-processing

Image Dataset Visualization

Here we have used Tensorboard web service to upload all our dataset and visualize it and plot it to learn some insights into the dataset, like outliers and what the distribution is.

The different color channels RGB were distributed along this way to detect outliers, we decided to go for pixel values less than 10 and more than 220 for RGB values in outlier detection.

Validation Techniques

1. For validation the validation\_split attribute is set to 0.2 hold-out validation for the initial model.  
2. Then we have moved on to using 5-fold cross validation but it's found to have no effect on the accuracy score.  
3. The final model validation is focused more on batch modulation than on splitting of training data.  
4. We have used ‘mean-square-error’ and ‘categorical\_crossentropy’ for validation, in the final model, both giving similar results, but since the dependent variable is not categorical, ‘cross entropy’ can’t be used.

Performance Measures

1. For the training scores, ‘accuracy’ and ‘loss’ are used to monitor the history of the model training.

2. Optimization is done through ‘adamax’ and activation function used is softmax.

3. The final results are not based on accuracy or other metrics, they are based on the levels on PSNR and SSIM in the final output images, discussed later.

4. A comparative report is done on the basis of these 2 measures between BM3D and our CNN model.

A small brief on PSNR and SSIM

1. 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.

2. The Structural Similarity Index (SSIM) index is a full reference metric; in other words, the measurement or prediction of image quality is based on an initial uncompressed or distortion-free image as reference.

3. Basically PSNR will guide it for better noise removal, and SSIM for feature preservation.

REFERENCES

1. Duan, Y., Wang, J., Kam, M., & Canny, J. (2005). Privacy preserving link analysis on dynamic weighted graph. *Computational & Mathematical Organization Theory*, *11*(2), 141-159.
2. Kerschbaum, F., & Schaad, A. (2008, October). Privacy-preserving social network analysis for criminal investigations. In *Proceedings of the 7th ACM workshop on Privacy in the electronic society* (pp. 9-14).
3. LeFevre, K., DeWitt, D. J., & Ramakrishnan, R. (2006, April). Mondrian multidimensional k-anonymity. In *22nd International conference on data engineering (ICDE'06)* (pp. 25-25). IEEE.
4. Sweeney, L. (2002). k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *10*(05), 557-570.
5. Liu, X., Lin, H., & Zhang, C. (2012). An Improved HITS Algorithm Based on Page-query Similarity and Page Popularity. *J. Comput.*, *7*(1), 130-134.
6. <http://pi.math.cornell.edu/~mec/Winter2009/RalucaRemus/Lecture4/lecture4.html>