Data Reduction and Deep-Learning Based Recovery for Geospatial Visualization and Satellite Imagery

A Thesis Submitted to the
College of Graduate and Postdoctoral Studies
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Jarin Tasnim

©Jarin Tasnim, December/2020. All rights reserved.
Unless otherwise noted, copyright of the material in this thesis belongs to the author.

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building
110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan
Canada
S7N 5C9

Or

Dean
College of Graduate and Postdoctoral Studies
University of Saskatchewan
116 Thorvaldson Building, 110 Science Place
Saskatoon, Saskatchewan S7N 5C9
Canada

Abstract

The storage, retrieval and distribution of data are some critical aspects of big data management. Data scientists and decision-makers often need to share large datasets and make decisions on archiving or deleting historical data to cope with resource constraints. As a consequence, there is an urgency of reducing the storage and transmission requirement.

A potential approach to mitigate such problems is to reduce big datasets into smaller ones, which will not only lower storage requirements but also allow light load transfer over the network. The high dimensional data often exhibit high repetitiveness and paradigm across different dimensions. Carefully prepared data by removing redundancies, along with a machine learning model capable of reconstructing the whole dataset from its reduced version, can improve the storage scalability, data transfer, and speed up the overall data management pipeline.

In this thesis, we explore some data reduction strategies for big datasets, while ensuring that the data can be transferred and used ubiquitously by all stakeholders, i.e., the entire dataset can be reconstructed with high quality whenever necessary. One of our data reduction strategies follows a straightforward uniform pattern, which guarantees a minimum of 75% data size reduction. We also propose a novel variance based reduction technique, which focuses on removing only redundant data and offers additional 1% to 2% deletion rate. We have adopted various traditional machine learning and deep learning approaches for high-quality reconstruction. We evaluated our pipelines with big geospatial data and satellite imageries. Among them, our deep learning approaches have performed very well both quantitatively and qualitatively with the capability of reconstructing high quality features. We also show how to leverage temporal data for better reconstruction. For uniform deletion, the reconstruction accuracy observed is as high as 98.75% on an average for spatial meteorological data (e.g., soil moisture and albedo), and 99.09% for satellite imagery. Pushing the deletion rate further by following variance based deletion method, the decrease in accuracy remains within 1% for spatial meteorological data and 7% for satellite imagery.

Acknowledgements

I would like to express my most sincere gratitude to my supervisor, Dr. Debajyoti Mondal, for all his patience and inspiring discussions. I would like to thank him for always leading the way and advising me in difficult situations that were impossible for me to overcome. I would like to thank other committee members: Dr. Ian Stavness and Dr. Lingling Jin, for their suggestions and comments on my research and thesis.

This work is supported by Global Water Future Grant. I would like to thank Dr. Kevin Schneider for his encouragement. I would like to thank Dr. Ian Stavness again for introducing me to the Plant Phenotyping and Imaging Research Centre (P2IRC) on deep learning.

My special gratitude belongs to my parents, who have made many sacrifices in their lives to make me happy. I can never repay what they have done for me.

I would like to thank my dear husband for always believing in me. I am grateful to him for encouraging and supporting all my dreams.

Lastly, I would like to thank the Almighty for everything I have.

This thesis is dedicated to my mother, Shamina Pervin, my father, Md. Jahangir Alam and my dear husband, Sakib Mostafa.

Contents

List of Tables

List of Figures

List of Abbreviations

GAN	Generative Adversarial Network
PCA	Principle Component Analysis
FPCA	Functional Principle Component Analysis
SVD	Singular Value Decomposition
VGG	Visual Geometry Group
MSE	Mean Squared Error
MAE	Mean Absolute Error
PSNR	Peak Signal to Noise Ratio
SRGAN	Super Resolution Generative Adversarial Network
WRF	Weather Research & Forecast
GPU	Graphics Processing Unit
DNN	Deep Neural Network
RNN	Recurrent Neural Network
SLC	Scan Line Corrector
CNN	Convolutional Neural Network
JPEG	Joint Photographic Experts Group
IDW	Inverse Distance Weighting
SNN	Shallow Neural Network
SSIM	Structural Similarity Index
SLC	Scan Line Corrector
PCA	Principle Component Analysis
RSS	Residual Sum of Squares
ReLU	Rectified Linear Units

Chapter 1

Introduction

In the era of big data, the collection, storage, and retrieval of data is a barrier to the researchers and decision-makers for fast manipulation and analysis. It is also challenging to work with big data in local computing machines with limited memory and slow internet access. Therefore, it is important to compress the big datasets into a smaller one, which will not only reduce storage requirements but also allow light load transfer. An efficient data compression approach, along with a learning-based model capable of reconstructing the whole dataset with maximum accuracy can minimize all these challenges and pave the way to smooth data analytics. In this chapter, the motivation behind this thesis will be explained in detail, followed by research questions to be answered, and finally, the key contributions.

1.1 Motivation

The revolutionary progress of sensors, IoT technology, and other scientific data-collecting devices, is giving rise to massive data repositories with high velocity. As a result, data-driven decision making is relying more on a combination of analytical and visualization techniques [60]. However, any processing on such immense data demands high-bandwidth, high-capacity storage, and high-performance computational power [70]. Most often, user quires on such data are executed on the server and the resulting query results (visual plots such as contour plots, heatmaps, and various statistics) are sent over a network [16]. Sometimes, the user queries for a data sample, and the sample is analyzed on the client-side. However, both of these techniques have limitations. While the server-client communication load imposes various limitations on user interactions, the data sample-based analysis is prone to various sampling artifacts. Data reduction is a common approach to deal with big data [98], where the main goal is to retain the most informative subset of the data and identify redundant subset to be discarded. Various forms of data reduction techniques such as outdated data deletion [101], lossless or lossy compression [74], dimensionality reduction [103], and discretization [51] are used in practice.

Datasets like geospatial weather data or remote sensing data have high resolution in space, time, and modality which makes it a challenge to applying conventional analytical and visualization methods. Although, this type of data is collected every day and at every point in the earth, the spatio-temporal data can be highly redundant [63]. That is why, such redundancy can be removed without significant loss in quality,

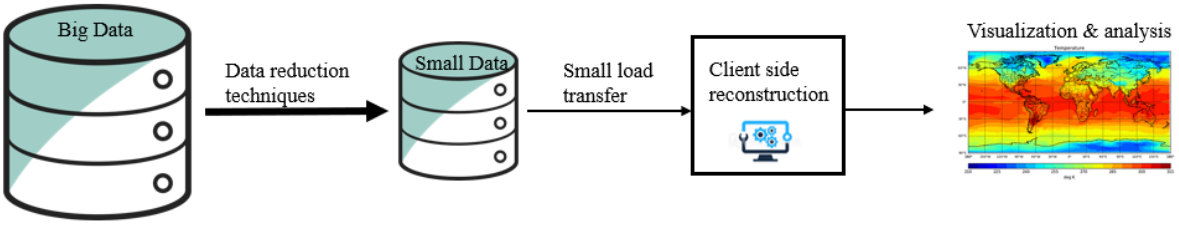


Figure 1.1: Data reduction and reconstruction overview.

preserving the structure of the data. Moreover, systematic and aggressive deletion of data allows easier spatial observation and analysis [42, 63].

In this thesis, we explore data reduction techniques coupled with machine learning based reconstruction. The motivation is to create a reduced dataset along with a machine learning model, which can be used on the client-side to produce visualizations that closely approximate the server-side (original) visualizations. Since small changes are not easily detectable by our eyes, for generating a visual picture, it is desirable to reduce the data in a way that has the least impact on the visual quality once recovered. One can compress the reduced dataset further by using image compression algorithms [98].

1.2 Research Questions

The main focus of this research is providing a pipeline for data reduction along with a reconstruction method for both geospatial data, plots, and satellite imagery. Firstly, developing different data deletion strategies that do not compromise with visualization quality when reconstructed. Secondly, building compact, consistent, and reliable machine learning based reconstruction models.

The growing rate of data generation requires scalable resources associated with increasing time and cost. The best way to deal with this problem is to reduce the data in such a way that will have a minimum loss of information along with a good amount of storage and bandwidth saving. Consequently, our plan is to design systems that delete data carefully and extensively. The idea for recovery of this deleted data is having a trained machine learning model in a local machine, capable of reconstructing the deleted data samples from the available compact dataset.

The thesis aims at answering the following specific research questions:

1. How to reduce data by deleting data samples yet retaining important information for subsequent recovery?
2. How to reconstruct missing data with high reconstruction quality?
3. What are the trade-offs behind this whole deletion-reconstruction pipeline?

1.2.1 Our approach

We are going to address the above mentioned questions in two steps.

Firstly, we will explore simple and straightforward uniform deletion schemes. We will analyze their capability in retaining visual information. Secondly, we will further push the deletion rate to the extreme but by adopting careful tactics that focus on more redundant information. The overall goal is to delete as much data as possible retaining important information.

We will explore a wide range of machine learning and deep learning models for the reconstruction of the deleted data. The machine learning algorithms cover Bayesian Ridge regression and shallow neural networks. For deep learning models, generative adversarial networks are taken into consideration for their notable performance in generating data distributions that are realistic and similar to the original data. We train generative models that can recover the full data from the reduced one. This part is mostly inspired by the advanced and established image processing methods of recovering missing data in two scenarios. The first one is creating super-resolution images and the second one is inpainting or filling in missing pixels. Our attempt is to develop these models in such a way that can retrieve the deleted data with high accuracy and minimum error. We intend to implement our data deletion strategies on two large spatio-temporal datasets, i.e. Satellite imagery and geospatial plots, in order to implement our data deletion strategies. The results are evaluated in two forms which are matrix and image for machine learning models. The evaluation metrics are explained in detail in section 5.1.3. The results are also compared with interpolation and JPEG compression techniques.

1.3 Contributions

In this paper, we explore data reduction and recovery techniques for geospatial contour plots and satellite imagery. Therefore, a large body of our work is inspired by various image processing techniques. Our contributions are as follows:

1. We examine two types of data deletion strategies: uniform deletions (checkerboard and grid pattern), and uniform deletion followed by a variance based deletion. The uniform deletion achieves a guaranteed 50% (checkerboard) or 75% (grid pattern) data reduction, with a high recovery rate. However, our variance based deletion technique can further improve the reduction rate over 76% on both weather datasets and satellite imagery. The quality of the reconstructed images does not decrease over 0.07 for Structural similarity index (SSIM) and 9.16 for Peak to signal ratio (PSNR). Here SSIM and PSNR are metrics for measuring the similarity between the original image and the reconstructed image. The details of the SSIM and PSNR are explained in Chapter 5.

2. For the reconstruction of the removed data, we adopt a wide range of machine learning (Bayesian Ridge Regression, Shallow Neural Network) and deep learning models (Super-resolution generative

adversarial, Inpainting network). With 75% data deletion, the models were able to reconstruct both weather datasets and Satellite imagery with high accuracy (above 98% SSIM and 34dB PSNR). The visual difference between the original and reconstructed data was the least for deep learning approaches. Among them, inpainting had outperformed all other methods qualitatively and quantitatively. The SRGAN and inpainting models are explained in Chapter 4.

3. While using deep learning approaches for reconstruction, we examined two techniques. One is to use the idea of constructing a higher-resolution image from lower-resolution images. Here we employed a Super Resolution (Generative Adversarial Neural Network) GAN model [53] together with a data imputation technique. The other idea is to use image inpainting, i.e., repairing damaged regions in an image. Traditional image inpainting network [107] often ends up reconstructing missing pixels in a blurry and imprecise way because of the lack of enough spatial information. Therefore, we modified the traditional image inpainting network [107] to incorporate information from a temporal or interpolated data. A temporal data is a plot or image of another timestamp from the same spatial region, and the interpolated data is a plot obtained by inpainting the image using interpolation. The modified inpainting network outperformed SRGAN model and had better accuracy and visual quality. To the best of our knowledge, this is the first data reduction work that uses the concept of temporal data for recovery in order to deal with resource constraints.

1.4 Chapter Organization

This document is structured as follows.

In Chapter 1, we have explained the motivation of this thesis work. We have explained the research problem, our approach to answering the research questions, and the outcome we were able to achieve.

In Chapter 2, the related works are explained in two sections. The first one discusses the data deletion concept and its use in different situations. The second one discusses the reconstruction methods covering both the spatial and spatio-temporal recovery techniques.

Chapter 3 provides a brief introduction about proposed data deletion strategies along with their justification. The data deletion strategies are divided into uniform deletion and variance based deletion.

Chapter 4 describes the reconstruction strategies that we have considered. The reconstruction methods that are considered are categorized into statistical, machine learning, and deep learning. Statistical methods include simple and widely used interpolation techniques. Machine learning includes Bayesian ridge regression and shallow neural network, where both works on features engineered from the data. Deep learning methods include two types of generative adversarial networks. We have explained the theoretical background, architectures, and loss functions for each of these reconstruction models.

Chapter 5 illustrates the experimental design of different data deletion and recovery techniques in detail along with an explanation about the data source and processing of datasets. We have also explained both

the qualitative and quantitative results from the experiments. We have compared the results from different data reconstruction models. We have also discussed the advantages of proposed deletion strategies over conventional compression techniques. The metrics used to compare the results are also discussed in this chapter.

Finally, Chapter 6 represents the concluding remarks of this research. Then, the main contributions of this thesis are summarized along with some limitations that can provide the direction of possible future works revealed as a result of this thesis.

1.5 Declaration

Throughout this document, the term "we" refers to the author and the reader following the theoretical computer science norm for scientific writing.

I confirm that I have conducted this research and written this document under the supervision of my advisor Dr. Debajyoti Mondal. Part of the results has been accepted to be published in proceedings of *2020 IEEE International Conference on Big Data (IEEE BigData 2020)* [89].

Chapter 2

Background Literature

In this chapter, the first section will discuss a range of widely used data reduction approaches in different scenarios. Data reduction is a prime solution towards solving the problem of resource limitation that comes with big data management. Most of the data deletion process either follows ordinary rules such as redundant data deletion or uses code designs such as Huffman coding. Only a few examples exist in the literature there that follow data driven solutions.

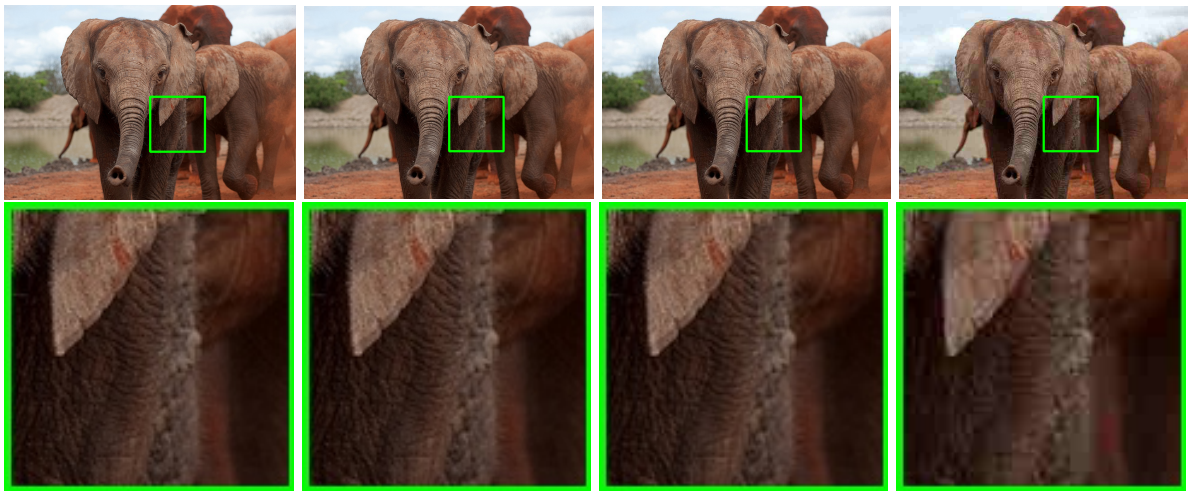
In the second section, previously used data reconstruction techniques for spatial datasets and spatio-temporal datasets will be introduced in two different sections. In each section, statistical and machine learning methods will be described. There are a considerable number of statistical methods available for data reconstructions. However, they often rely on local neighborhood based analysis and fail to generate or recover missing data by understanding the global context. This issue can be solved by leveraging machine learning methods with having automatic-feature selection procedures.

2.1 Data Reduction Methods

Data reduction is a common approach to deal with big data [98], where the main goal is to remove duplicate, uninformative, easily retrievable data points from the data that does not compromise with data quality.

Unnecessary Data deletion: There are different states of data that can allow it to be deleted for reducing storage requirement, e.g., outdated data which are no longer required or duplicated data. Growing information is often handled by either passive or active strategies. In the passive strategy, the data is saved as backup and not allowed to be deleted until it got completely valueless. In the active strategy, the data points are immediately deleted when it became obsolete [34].

A technique of optimizing storage utilization is *deduplication* which allows removing all the duplicates. Generally, the deduplication process identifies and store unique chunks of data while other chunks are compared with the stored ones and given a reference that points to the matched. However, the building of a reference counter is space consuming. Przemyslaw et al. [88] proposed a deletion algorithm for scalable and content-addressable storage with global deduplication that can result in deletion ranging from 5% to 30% by identifying the requirements of deletion and satisfies them by allowing deduplication while deletion.



(a) Original Image **(b)** 30% compression **(c)** 60% compression **(d)** 90% compression

Figure 2.1: First row shows example of lossy image compression. Second row shows zoomed in sections.

Compression: *Image compression* methods eliminate redundant information from a file, and thus saves memory space. Usually, there are two forms of compression: lossless and lossy. As the name implies, *lossless* compression guarantees 100% restoration after decompression. Two popular lossless compression techniques are TIFF and PNG [6]. They are used in situations, where the loss of information is not acceptable such as medical-imaging [85].

On the contrary, *lossy compression* removes useless or low level information from the file which allows a decline in storage requirement. The recovered data in lossy compression does match entirely with the original data. The goal is to keep the recovered image after decompression nearly the same as the original image by visual comparison. Widely used lossy compression techniques are JPEG and GIF [79].

Lossless compression typically offers a 33% to 60% reduction of size. But depending on the data, lossy compression can add to this number by 95% [14]. As shown in Fig. 2.1, the more the compression rate, the poor the quality of the image. This massive amount of reduction is often undesirable. JPEG compression algorithm has four steps which are RGB to YCbCr color space conversion, Discrete Cosine Transformation, coefficient quantization, lossless encoding.

Dimensionality Reduction: *Curse of Dimensionality* causes issues while analyzing and organizing data with complexity and high dimension [9]. This is addressed by *dimensionalty reduction* [96] methods which generate new low dimension features rather than eradicating low level information. This new set of features are either a low dimensional representation of data or a weighted combination of the existing data. This is done by applying some projection matrix to the original data. The lower dimensional data is then transformed back using the basis vectors, which are the principal components of the original data. There are various kinds of dimensionality reduction techniques depending on the criteria it follows while reducing the dimensions.

First, the linearity or nonlinearity used in creating new data. Second, the covariance matrix is taken into consideration so that the new reduced data maximizes the representation capability. Third, the number of variables considered in reduction methods [15].

Dimensionality reduction is a widely used approach to cope with the data size. Principle component analysis (PCA) and singular value decomposition (SVD) are often used to reduce the dimension of the data maintaining the original characteristics of the data while reconstructing [22,64]. *Principle component analysis* (PCA) maximizes the variance in low dimension while mapping the high dimensional data. The eigenvectors and eigenvalues are calculated from the covariance matrix of the data. The eigenvectors are ranked according to eigenvalues and the projection matrix is constructed from the top k principle components or highest eigenvalues. This projection matrix transforms the high dimensional data into k lower dimensional new data. Later on, these eigenvectors with the largest eigenvalues are used to reconstruct the data capturing a large amount of data variance. t-SNE or t-distributed stochastic neighborhood embedding is also a non-linear dimensionality reduction algorithm [96]. PCA tries to preserve the global variance of the data whereas, t-SNE tries to preserve the local variance of a certain point.

Discretization: *Discretization* is the process of transforming continuous valued data into discrete ones which help in reducing storage usage. It divides the range of the continuous data into a smaller number of sub ranges and stores the data as qualitative discrete values. It is always easier to understand the data in discrete form rather than continuous, as continuity may result in infinite degrees of freedom and also complex non-linearity. Moreover, the storage requirement also gets reduced. However, there has to be a balance between the loss of information and the number of sub-ranges to create discrete data as less number of sub-ranges causes increased loss and vice versa [51]. Example of discretization methods are k-means, Decision Trees, equal-width discretization [58].

Selection masks: *Selection mask* is a binary mask composed of zeroes and ones where 1 denotes selected and vice versa. However, their framework is quite a dependant on the task which is classification. Ohemcke et al. [69] proposed a learnable selection mask that can automatically select only the relevant part of the input image using a neural network that does not compromise with the machine learning model's performance.

2.2 Reconstruction Methods

In this section, we review the literature on the data reconstruction techniques that fill in missing or corrupted data. Depending on the use of data as features, the recovery techniques are explained in two categories: spatial and spatio-temporal. In each category, statistical and machine learning methods are explained.

2.2.1 Spatial Dataset

Statistical Reconstruction Techniques:

Imputation: One of the most common ways to deal with missing data is the deletion of the samples with missing data. Although this is straightforward and simple to implement, discarding samples with missing data introduces bias or misrepresentation in the analysis. Rather than completely removing a full sample, a simple statistical way of reconstructing the missing data is imputation [95]. *Imputation* refers to replacing missing data with substituted value or an estimate. Missing data can be replaced with mean, mode, or most frequent value of the dataset. However, they often do not capture the spatial correlation, as well as introduce bias in the data. *Regression imputation* predicts the missing variable depending on the other variables using a linear equation, as follows:

$$\hat{\mathbf{y}} = \hat{\beta}_0 + \hat{\beta}_1(\mathbf{x}) \tag{2.1}$$

Here, $\hat{\mathbf{y}}$ is the variable with missing values and $\mathbf{x}$ is the known variable, and $\hat{\beta}_0$ and $\hat{\beta}_1$ are the regression coefficients. But this does not have an error term introduced in the estimation. As a result, the estimates seem to fit perfectly with the data and thus fail to capture the uncertainty. This issue can be addressed by introducing error term but this causes more error in the estimate than general by adding complexity to the model [29].

Interpolation: Filling in missing regions of an image based on its surrounding context is considered as *interpolation* problem [8]. The most commonly used interpolation methods are the nearest neighborhood interpolation, bilinear interpolation, cubic convolution interpolation, Krigging Interpolation [41, 99].

Optimal interpolation techniques are often used in recovering missing data to preserve visual quality in geospatial data. Pang et al. [13, 72] compared Shepard's inverse distance weighting interpolation and multiquadric interpolation to reconstruct missing geospatial data and integrates different parameters such as uncertainty into visualization for ensuring accuracy. Details of Shepard's and multiquadric interpolation is given in section 4.1. We also examined these approaches in our experimental analysis.

Machine Learning Approaches:

Machine learning models use statistics to learn complex patterns in massive datasets in different tasks such as classification and regression. One of the simply developed and interpretable machine learning models is the regression model, which is capable of estimating the relation between dependent and independent variables, where independent variables are usually features or covariates. To estimate more complex and stupendous patterns, the regression model often requires intelligent feature engineering by humans which is inconvenient while working with large multidimensional data. However, in such scenarios, deep learning models designed to process data incrementally in different layers, are found to be very effective in extracting high level features or abstract representation from data without human analysis [33]. Especially, autoencoder architectures can

learn compact data representation similar to PCA, but more effectively. In image compression, a variety of deep learning models, e.g. convolution neural network [7,54], recurrent neural network [44,93,94], probability models are deployed [65]. *Neuron* in artificial intelligence, is a simulation of brain neuron, which takes data as input and learns from it in order to predict the desired outcome in the output. Neuron typically contains weights which are adjusted as the learning proceeds in order to model the relation between input and output. It takes input values as numbers and then multiplies them with weights and adds biases. The weighted sum is multiplied by the activation function. Usually, *artificial neural network* (ANN) consists of three layers, which are the input layer, hidden layer, and output layer. The hidden layer consists of a number of neurons. A neural network is called *deep neural network* (DNN) when there are many hidden layers in it [3]. *Recurrent Neural Network* (RNN) is a type of neural network that learns temporal behavior of data by having connections between neurons along the temporal sequence [78]. *Autoencoder* transforms input data into a lower dimension and then reconstructs the input back. It is built of two networks which are encoder and decoder. PCA offers the same type of representation by linear transformation whereas autoencoders can be both linear and non-linear. More about autoencoders are explained in Section 4.2.3.

Autoencoders are also applied for data compression in different areas [56, 90, 91]. For example, Diao et al. [23] used a recurrent autoencoder in a distributed system to compress images from a set of distributed sources. Pan et al. [70] designed autoencoders for 3D time-series data along with a coding and decoding block to achieve binary representations in a compressed format. Diao et al. [24] explained a distributed image compression technique based on a group of distributed source by designing a recurrent autoencoder. Chandak et al. [17] proposed a compressor that is based on a prediction-quantization-entropy coder framework where they have implied least square and neural network models. Feng et al. [43] designed two separate network architectures, one of which compresses the data, and the other one reconstructs it. Both of them are optimized following a unified loss function. Marchetti et al. [63] adopted an adaptive spatial dispersion clustering that reduces the size of spatial data by creating spatial clusters maximizing spatial correlation keeping the loss minimum. The goal was to generate new data that can be used for spatial prediction using kriging methods. Doutsi et al. [27] introduced a neuro-inspired compression mechanism of RGB images that is entirely inspired by the neuroscience behind the filtering and compression mechanism of the human brain. Their method goes through a set of steps which are RGB to YCbCr transformation, RIF coefficient shifting, and color channel downsampling which help greatly in enhancing the results. There are different deep learning architectures that are capable of dealing with recovering missing information successfully in the domain of image inpainting. The very first use of convolutional layers in image inpainting was by Xie et al [102]. The results were further developed and taken to an aesthetic level when generative adversarial networks were incorporated into the task of image inpainting [75]. *Generative adversarial network* comprises of two networks which are generator and discriminator network. The generator generates data while the discriminator tells if the data is original or generated by the generator model. The generator tries to minimize the loss between the original and its generated data when the discriminator tries to get better at discriminating by maximizing the loss. This

loss is *adversarial loss*. GAN is described in more detail in Section 4.3. GAN was later improved by global and local wgan network by Yu et al [107]. This model is one of the foundations for this work. Irregular holes and complex structures are always tricky to fill in. Nazeri et al. [66] proposed a two stage adversarial network, where the first stage approximates the edges in the missing area and the second one fills in the hole. There exist many studies that focus more on irregular holes in images and developed models with partial convolution [57, 104, 108].

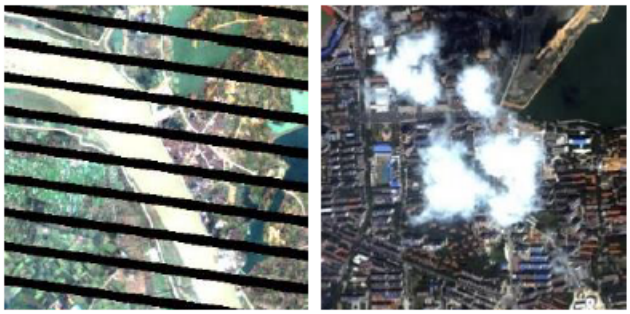


Figure 2.2: (left) Landsat ETM+ SLC off. (right) Image with thick cloud cover (adapted from [112]).

2.2.2 Spatio-Temporal Recovery

Super-resolution is the task of recovering the high resolution image data from low resolution image data [100]. This task is too sensitive to be done by interpolation methods as there needs to be a proper construction of finer textures. As a result, learning based algorithms are explored in this field, where the transformation between low and high resolution data is learned. SRCNN is a three layer convolution neural network which shows better performance than interpolation [25]. Later, a special perceptual loss based GAN model is proposed, which is capable of bringing finer and more realistic high resolution image data [53].

Spatial information provides complementary information for reconstructing deleted locations. But it sometimes lags when it comes to complex structures. For instance, if the area inside the black square is unknown as shown in Fig. 2.3, most of the spatial regions do not offer enough information to construct the structure, and so cause a failure. To solve this problem, the dataset needs to be extremely rich with such examples so that the model can learn to reconstruct accurately. In such scenarios, temporal information at the same location can contribute to the information about the shapes to be reconstructed. We can see in Fig. 2.4, part of the temporal images are the same in either color or structure. However, there can be mismatches in color due to time difference, weather as shown in the third row of Fig. 2.4. Formerly, temporal data has been leveraged to reconstruct unknown information in several studies. For example, satellite images often have missing information caused by cloud obstacles or sensor failure due to scan line corrector(SLC) off as shown in Fig. 2.2. All these missing information are best recovered when a combination of spatial information and temporal or spectral information is used [81].

Figure 2.3: (left) Original Image. (middle) The black region shows missing pixels. (right) Image produced by an inpainting algorithm.

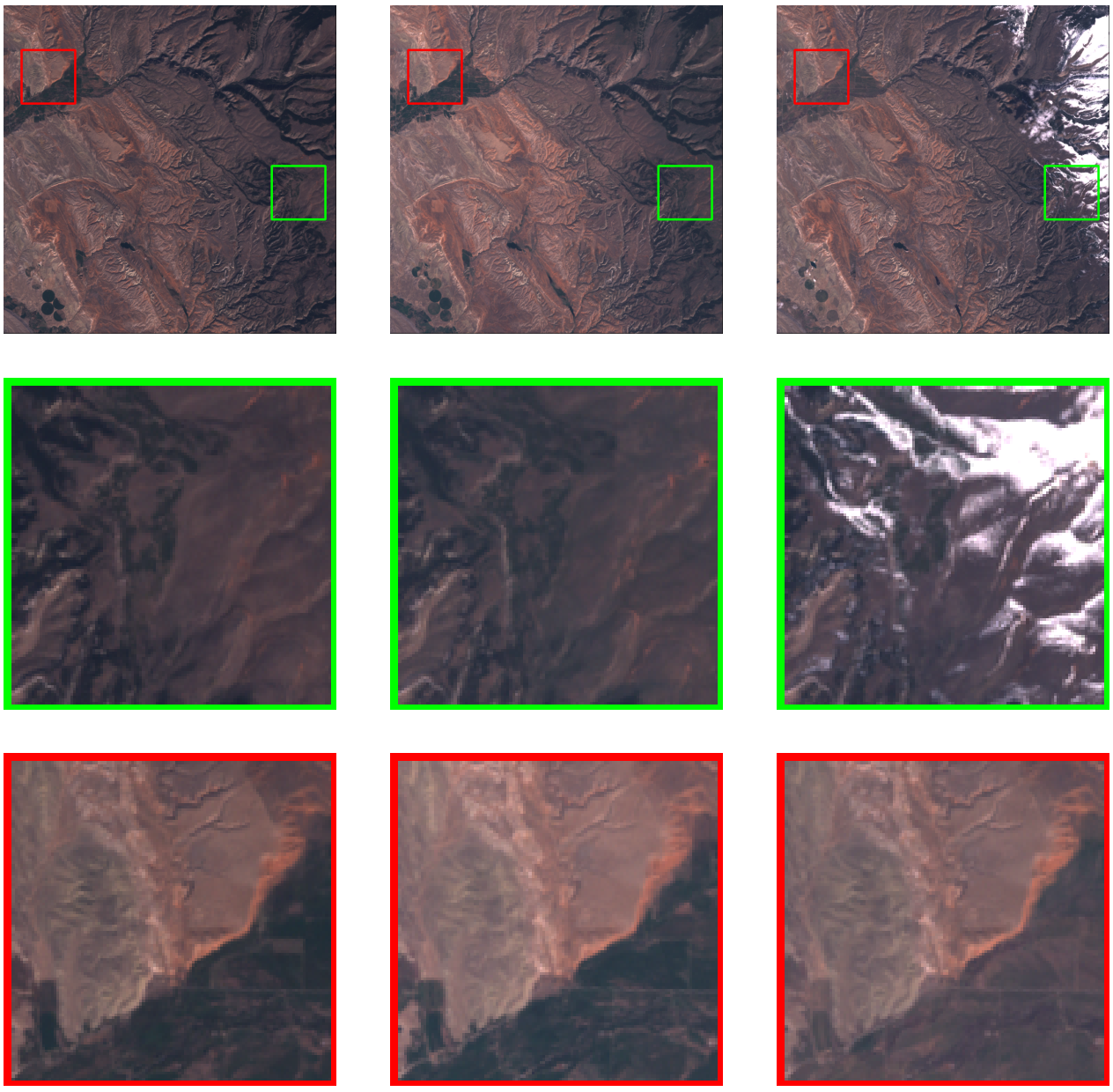
Statistical Reconstruction Techniques: The classical method of using temporal data as supplementary radar information is replacing the positions that are missing with that of the temporal ones. There are two ways to do this: direct and indirect replacement by temporal data. If there is little difference with the temporal data then the missing positions can be directly filled with the temporal data. Usually, this scenario is rare. Therefore, the temporal data needs to undergo some transformation before filling in the missing data.

Since satellite datasets are not aligned, a local linear histogram matching algorithm is often applied to the temporal data to match the histogram of the missing data. Although this is quite a simple method, it matches the histogram in the local neighborhood. Therefore, it may become really sensitive. Moreover, due to the dependency on the local histogram, it fails in generating smaller features such as texture [4, 80].

Regression is another way to transform the temporal data, where the data values work as features. So Zeng et al. [110] proposed a weighted linear regression model of the temporal data that can help in minimizing the temporal differences. The regression equation is solved by the least square method. To cope with the unavailability of proper temporal data, they have incorporated a non-reference based regularization method.

Functional principle component (FCPA) analysis has varied applications in multivariate dimensionality reduction for time series datasets [45]. This method includes PCA as a step but considers the dimension order of the multivariate data while creating a low dimensional representation. The time or space order can not be permuted like PCA. PCA usually captures the overall variance of the data. But time series data are often too complex to detect the variation. This issue is overcome by FCPA as it can capture the variance with time [97, 109].

Machine Learning Approaches: STS-CNN [112] takes three inputs, which are data with unknown information, temporal data (with no unknown information), and data that has its unknown information replaced by the information using temporal data, as shown in Fig. 2.5. These inputs are fed to a specially designed convolution neural network that extracts features from all the inputs. Dilated convolution layers are used to bring features from a wide viewpoint. But these models fail when the temporal data is sparsely spaced [112]. Another work uses convolutional layers to build three separate networks that extract features content-wise,



(a) Image date 22-09-2001 **(b)** Image date 23-07-2002 **(c)** Image date 21-04-2003

Figure 2.4: First row shows three satellite images at the same location in different timestamps. Second and third row show zoomed in sections.

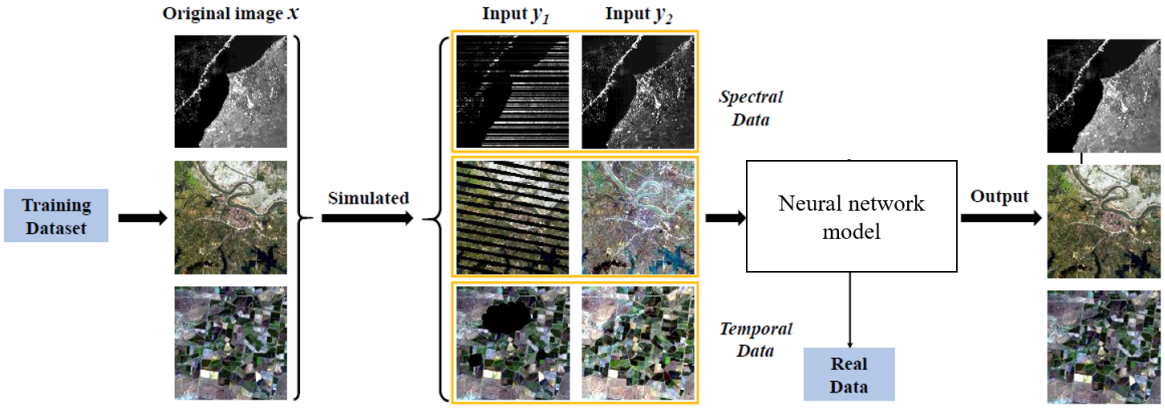


Figure 2.5: Explanation of the inputs by Zhang et al. (figure adapted from [112]).

spectral-wise, and texture-wise. The adversarial loss function used to train the network is the mean squared error between temporal data and the data with missing information [18].

Cheng et al. [19] have proposed a two-step method for reconstructing the missing information of 2D spatio-temporal data. The reconstruction method incorporates an inverse distance weighting interpolation method for obtaining some coarse result, followed by another interpolation method for obtaining a fine result with the help of spatial and temporal information. Finally, the data samples with finely interpolated missing locations are fed to a neural network model.

Dong et al. [26] trained a DCGAN to generate SST images in an unsupervised way. Later, the trained generator is used to train the inpainting network, which learns the representation of the original image in a low dimension with a new loss function.

Spatio-temporal features are widely used to fill in any hole or missing region in video inpainting. Kim et al. [48] has proposed a recurrent temporal aggregation network that has an autoencoder structure. The input to the encoder is the reference temporal frames and the output is the extracted features which are summed with recurrent feedback and fed to the decoder network.

Zeng et al. [111] designed a spatial-temporal transformer network which is trained by an adversarial network. This network looks for similar frames in both neighboring and distant frames to better build a complex notion. Aidini et al. [2] proposed tensor based compression by learning a dictionary of the tensor data based on an Alternating Direction Method of Multipliers. By tensors, they refer to a 4D object with two spatial axes, one temporal and one spectral axis.

2.3 Summary

In this chapter, we have described related background literature for both data deletion and recovery. Although data deletion has been applied in different fields taking different names, there is no application of aggressive data deletion in order to solve resource problems associated with geospatial datasets and satellite

imageries. Therefore, data deletion methods are worth exploring that can preserve important information as well as guarantees a large amount of deletion. On the other side, there is a wide range of recovery methods available in different scenarios ranging from basic statistics to deep learning. Statistical methods often fail at reconstructing high level details. In the field of image processing, there are some promising techniques available that ensure good quality not only from the numerical aspect but also from the visual aspect. Therefore, our aim is to explore advanced image processing algorithms along with deep learning approaches to obtain high quality reconstruction of the deleted data.

Chapter 3

Data Deletion Strategies

In this chapter, we describe the deletion methods we used in our work. We explore two types of deletion approaches, which are uniform deletion and variance based deletion. Uniform row-column deletion in grids is capable of ensuring 75% deletion, whereas a checkerboard deletion ensures 50%. Our variance based deletion approach improves the deletion rate by over 75% depending on the data variance. The later one also ensures careful deletion preserving any shape changes in the data or the shape structures in the image. We only focus on image data for this work.

3.1 Deletion Strategies

3.1.1 Uniform Deletion

Uniform deletion refers to deleting samples following an orderly fashion, which means that deleted samples have uniform intervals between each other. This concept is also used in systematic sampling [12] where the first sample is picked randomly and the rest of the samples follow a fixed pattern. But in our uniform deletion, the first sample position is also fixed. So unlike systematic sampling, the mean and variance are determinable using $\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$ and $s^2 = \frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n-1}$ respectively, where n is the number of samples out of N are kept in the dataset. We examine two types of uniform deletions: Grid and Checkerboard deletion.

Grid Deletion:

Let M be an $m \times n$ matrix. If half of the rows and half of the columns are deleted uniformly, then the total number of data points to be deleted is $(\frac{m}{2} + \frac{n}{2} - \frac{mn}{4})$, as shown in Fig. 3.1. This implies a deletion of three-fourth of the data (e.g., see Fig. 3.1). If we remove the data samples regularly, then it is highly likely that the removed portion of the dataset is similar to the remaining data in statistical properties, geometrical structures, and textural components, whereas it also guarantees a 75% reduction of the dataset.

Checkerboard Deletion:

Although uniform deletion results in 75% deletion of data samples, it might cause less accuracy in reconstruction. To determine the trade-off, we also examined the effect of a low reduction rate. In particular, we

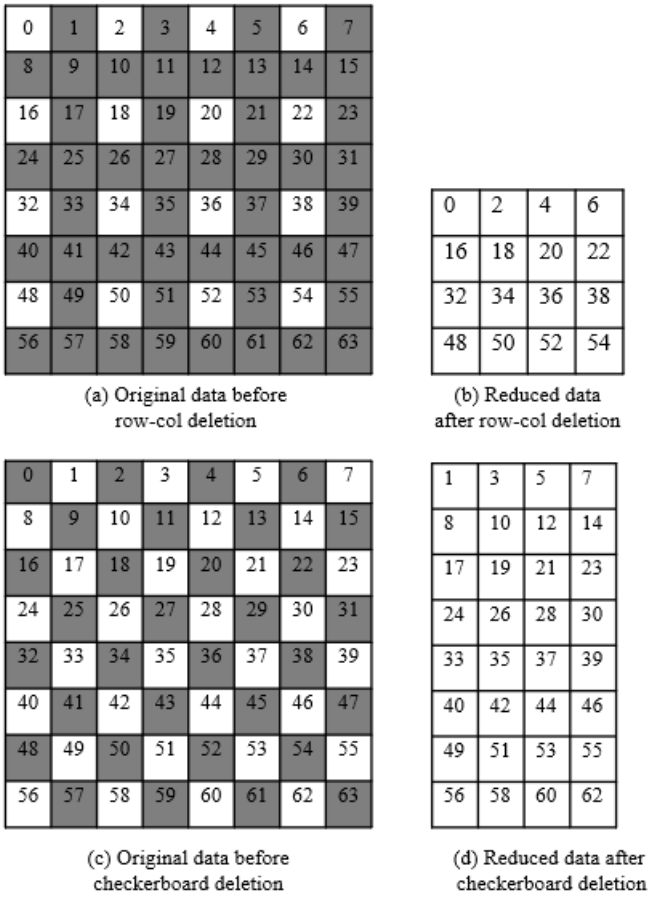


Figure 3.1: Illustration of grid deletion.

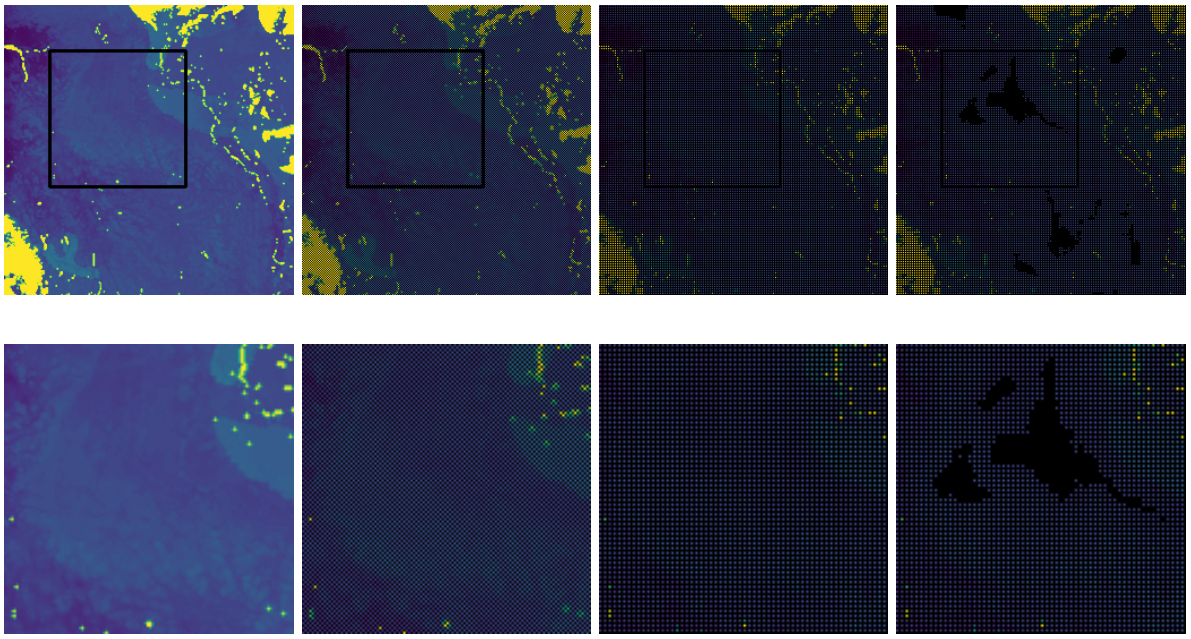
examined a deletion technique based on a checkerboard pattern that deletes half of the data samples, i.e., the number of matrix entries get reduced from $(m \times n)$ into $(m \times n)/2$ as shown in Fig 3.1, leading to a 50% data reduction.

3.1.2 Variance Based Deletion

Grid deletion offers quite a large amount of storage savings. However, the reduced data may still have some areas containing blocks of similar data points, which can be easily removed without reducing reconstruction accuracy.

Therefore, we propose a novel variance based method for increasing the reduction rate by deleting the easily retrievable data samples without compromising reconstruction accuracy. This method first deletes samples in the grid resulting in 75% deletion as in Fig. 3.2(c). Further, following our variance based deletion method, finds more removable data samples and increases the deletion rate, as shown in Fig. 3.2(d).

We first describe the variance based deletion technique and then provide an estimate of the number of



(a) Original Image **(b)** Checkerboard deletion **(c)** Grid deletion **(d)** Variance based deletion

Figure 3.2: Reconstruction results for satellite imagery.

additional data points that this method may detect for further deletion. Before we begin the whole method, we explain some of the terminologies.

Variance (σ^2) is the measure of how far samples vary from its average value, which is defined by the squared deviation of a random variable from its mean value. The more varied the data, the larger the variance is. *Percentile* implies the value under which a given percentage of samples in a data falls. The *nth* percentile of data is the value at which n percent of the data is below it. *Sample variance*, as the name implies, is the variance of the samples which is denoted by $\hat{\sigma}^2$. For the variance based deletion, we consider sample variance where the samples are picked from the neighborhood. For a particular sample, *Neighborhood* is the $k \times k$ field around it.

We first consider the sampling distribution of sample variances. For each location, we first compute the sample variance of a local k-neighborhood around that location. Let R be a threshold, which is the percentile measure of the sample variances. We define a data point to be a candidate for further deletion if its associated sample variance is less than R (assuming that they are easily recoverable). However, if all these data points with low neighborhood variance are deleted, then there may not always be enough information preserved to recover the shape boundaries as shown in Fig. 3.3. Therefore, such a straightforward deletion may not always be approximated effectively in the reconstruction phase.

To cope with such a problem, we propose a deletion strategy based on the variance of the neighborhood sample variance. This preserves the shape boundary to an extent that can be better reconstructed using the reconstruction techniques. For the variance of neighborhood sample variance, we define another threshold Q

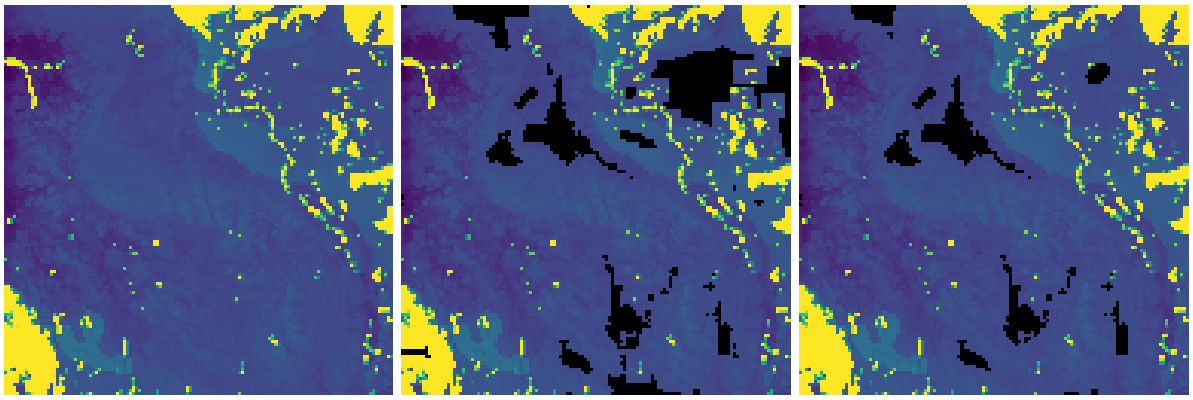


Figure 3.3: (left) Generated image after the grid deletion, from 25% of data points. (middle) One step variance based deletion. (right) Two step variance based deletion.

which is the percentile measure of the sample variance of the neighborhood sample variances. Our two step variance based deletion approach deletes every candidate data point, where the variance of the neighborhood variance is less than Q. In other words, every data point with $s^2 < R$ and $var(s^2) < Q$ is deleted, where s^2 is the neighborhood variance. Mask is a matrix having the same dimension as the original data, which has zero value indicating deleted positions and one value indicating non-deleted positions of the original data. Fig. 3.3(right) shows the mask after two step deletion where the deletion rate is lower, but the reconstruction will be more accurate.

To obtain an estimate of the potential data reduction, consider a normal distribution with mean μ and standard deviation σ. Then the expectation of sample variance is σ^2 [50]. Therefore, by Markov's inequality, the probability that the sample variance is smaller than R is

$$P[s^2 \leq R] \leq \left(1 - \frac{\sigma^2}{R}\right)$$

We run a simulation with a normal distribution to examine how the thresholds affect the number of data points satisfying various threshold conditions. We generate one dimensional normal distribution by keeping the mean and variance close to that of the datasets that we used in our experiments. We apply our variance based deletion approach with a range of possible values of k, R, and Q. For one dimensional data, neighbourhood area k is defined by the k number of data samples on both sides of a particular data sample. The deletion rates are reported in percentage in Table 3.1. More details of the Table 3.1 is provided in Appendix B. From the table, we can observe the following:

1. If we delete data samples with higher R and higher Q, the deletion rate is higher. This may remove important visual information that may not be recovered with high accuracy. Therefore, higher R and higher Q can be chosen if higher deletion is desired.

2. If data samples that satisfy $s^2 < R$ and $var(s^2) < Q$ are deleted, then the deletion rate is much lower than that with $s^2 < R$ or $var(s^2) < Q$ independently. But, the former ensures precise deletion along

Table 3.1: An experimental estimate for data reduction with a two step variance based deletion (red indicates a larger deletion).

Properties of the distribution			$\mu, \sigma = 4.45, 2.93$			$\mu, \sigma = 4.45, 7$		
k-Neighb.	R	Q	$s^2<R$	$V(s^2)<Q$	$s^2<R$ & $V(s^2)<Q$	$s^2<R$	$V(s^2)<Q$	$s^2<R$ & $V(s^2)<Q$
3	20	20	0.25	0.2	0.06	0.21	0.2	0.05
		40	0.25	0.4	0.11	0.21	0.4	0.1
		60	0.25	0.6	0.15	0.21	0.6	0.15
	40	20	0.41	0.2	0.1	0.4	0.2	0.1
		40	0.41	0.4	0.16	0.4	0.4	0.19
		60	0.41	0.6	0.24	0.4	0.6	0.28
	60	20	0.6	0.2	0.13	0.6	0.2	0.16
		40	0.6	0.4	0.27	0.6	0.4	0.31
		60	0.6	0.6	0.37	0.6	0.6	0.44
5	20	20	0.22	0.2	0.03	0.2	0.2	0.08
		40	0.22	0.4	0.08	0.2	0.4	0.1
		60	0.22	0.6	0.14	0.2	0.6	0.15
	40	20	0.4	0.2	0.04	0.4	0.2	0.13
		40	0.4	0.4	0.15	0.4	0.4	0.19
		60	0.4	0.6	0.26	0.4	0.6	0.28
	60	20	0.6	0.2	0.1	0.6	0.2	0.14
		40	0.6	0.4	0.23	0.6	0.4	0.25
		60	0.6	0.6	0.38	0.6	0.6	0.4
7	20	20	0.2	0.2	0.06	0.2	0.2	0.1
		40	0.2	0.4	0.11	0.2	0.4	0.14
		60	0.2	0.6	0.13	0.2	0.6	0.14
	40	20	0.42	0.2	0.1	0.4	0.2	0.13
		40	0.42	0.4	0.17	0.4	0.4	0.22
		60	0.42	0.6	0.29	0.4	0.6	0.24
	60	20	0.6	0.2	0.14	0.6	0.2	0.18
		40	0.6	0.4	0.28	0.6	0.4	0.3
		60	0.6	0.6	0.42	0.6	0.6	0.37

the edges when compared with the later two approaches. Fig. 3.3 illustrates a comparison between the two step deletion and a one step deletion.

3. Even with a cautious threshold choice, i.e., $R = 20$ and $Q = 20$, we have 5% data points satisfying the threshold conditions.

Since we have 25% data points remaining after the grid deletion, Choosing both R and Q as 20, creates a potential for deleting 1.25% additional data points, and the data reduction rate may increase to 76.25%. As we are examining all the data points, the time complexity of this algorithm is $O(n)$, where n is the number of data points and k is a fixed constant.

3.2 Summary

This chapter demonstrates the data deletion methods. The grid deletion is put forward over other deletion methods as it guarantees a 75% deletion rate that is a considerable number for the reduction of required resources. It is also able to keep the structure of the objects in the reduced data. Therefore, the reduced data can easily be used for visual analysis. However, the variance based method pushes the deletion rate beyond 75% and thus makes the overall deletion rate higher. This can greatly influence bigger data sets when a small amount of deletion saves resources significantly.

Chapter 4

Reconstruction Methods

A large amount of data deletion may be allowed if a high reconstruction accuracy can be achieved. In this section, we propose effective reconstruction strategies for good quality reconstruction of the geospatial plots and Satellite imagery. Firstly, a few interpolation algorithms are inspired by the work of Pang et al. [72], where they refer to sophisticated interpolation techniques for filling in missing or noisy data points in the geospatial dataset. Secondly, a regression algorithm is employed which is often capable of outperforming interpolation [67]. Thirdly, a generative adversarial network (GAN) is applied for the reconstruction task. Two different types of GANs are adopted, which are SRGAN and GAN based inpainting network.

4.1 Interpolation

Interpolation has been a popular way of filling in the unknown areas of geospatial data. We examined two extensively used interpolation algorithms, which are capable of modeling complex shapes: modified Shepard's inverse distance weighting [83] and Multiquadric interpolation [72].

4.1.1 Modified Shepard's inverse distance weighting interpolation

Inverse distance weighting (IDW) interpolation determines the unknown points from the weighted average of the values at known points. The interpolated value u at a given point x is calculated by,

$$u(\mathbf{x}) = \begin{cases} \dfrac{\sum_{i=1}^{N} w_i(\mathbf{x}) u_i}{\sum_{i=1}^{N} w_i(\mathbf{x})} & \text{if } d(\mathbf{x}, \mathbf{x}_i) \neq 0 \text{ for all } i, \\ u_i & \text{if } d(\mathbf{x}, \mathbf{x}_i) = 0 \text{ for some } i \end{cases} \tag{4.1}$$

where $w_i = (1/d(\mathbf{x}, \mathbf{x}_i))^P$ is the inverse distance weighting function, u is the interpolated value at $\mathbf{x}$, u_i is the known value at $\mathbf{x}_i$, d is the Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}$, N is the total number of points, and P is the power parameter. Intuitively, the greater the distance from the interpolated point, the smaller the weight or influence is. P also adds great influence on the nearest points. For this experiment, P is chosen to be 2 [83]. This incorporation of P is known as Shepard's IDW interpolation. The quality introduced in the interpolated data by this method is often too high, which means that the interpolation performs well

only on the available data but will fail in case of unknown and uncertain data like overfitting. Therefore, a modification is proposed to introduce uncertainty in the data [72].

A modified version of this interpolation method [83] uses only the points in the nearest neighbors in a sphere having k and thus the weights are modified as follows:

$$w_x = \left(max \left(0, \frac{k - d(\mathbf{x}, \mathbf{x}_k)}{k d(\mathbf{x}, \mathbf{x}_k)} \right) \right)^2 \tag{4.2}$$

Here, d is the Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}$. The implication of introducing k-sphere is the data quality becomes zero out of the sphere. Thus no noise can be added due to samples outside of the k-sphere.

4.1.2 Multiquadric interpolation

Polynomial function is function, which has positive integer powers of an independent variable in its equation. *Radial basis function* is a real valued function, which maps the input to a value, which is defined based on the distance from the input to a fixed center point. *Multiquadric interpolation* is based on modeling data using a function [72, 76]. The function, being radial basis function, has advantages over a polynomial function, as they are capable of capturing complicated shapes in topographical surfaces [20]. Multiquadric interpolation is defined by sum of weighted radial hyperbolic functions as follows.

$$f_x = \sum_{i=1}^{N} \alpha_i Q(x, x_i) \tag{4.3}$$

where Q is the radial basis function $Q(x, x_i) = \sqrt{(x - x_i)^2 + c^2}$, and c is the parameter that defines the shape of the interpolation surface, N is the total number of datapoints. The parameter c causes the interpolation to be flatter. The parameter c is kept 1 [72]. The coefficients α_i are calculated by solving a set of linear equations, representing N interpolation conditions which are:

$$f_{x_j} = \sum_{i=1}^{N} \alpha_i Q(x_j, x_i) \tag{4.4}$$

4.2 Machine learning Algorithm

We examined two regression models which are described in this section.

4.2.1 Feature engineering

The feature extraction method is based on the concept that nearer available data points are more correlated with the data to be estimated. On this ground, for every removed data sample, we look for available data samples in its $k \times k$ neighborhood and define this as the feature vector. Here k is varied for 3 and 5 to observe the effect of an increasing number of input features to the model. Due to removing the data samples in

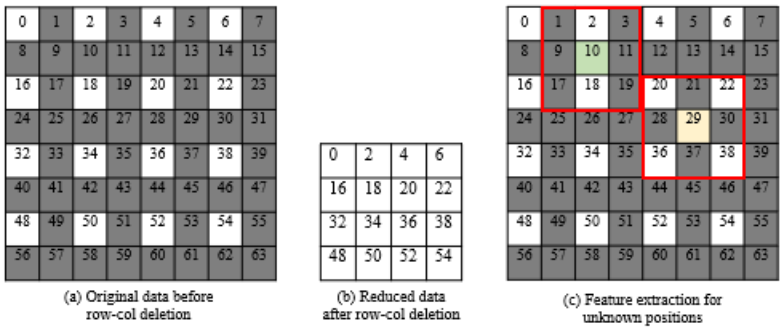


Figure 4.1: Feature extraction process for grid deletion. Red area shows the 3×3 neighbourhood where features are searched. White cells denote available ones and grey denotes deleted positions. For cells such as cell no 10 (highlighted in green), the number of available features is 2 and for cells like cell no 29 (highlighted in yellow) the number of available features is 4.

grids, for every other data sample, there are less than $k \times k$ data samples available in the grid which results in an imbalanced feature set to be used as shown in Fig. 4.1. However, the regression algorithm takes a fixed number of features. These problems are dealt with as follows: First, the maximum number of available features in a $k \times k$ neighborhood is assigned as the feature number k_w for the regression algorithm. Second, if any deleted data sample has less than k_w number of available samples in the neighborhood, then the rest of the data samples are imputed using mean values of the available ones. Total number of features for $k = 3$ is 4 and $k = 5$ is 6.

To mitigate the effect of noise, a new set of features are introduced for which the available data is passed through a Gaussian filter with σ standard deviation. Then the same number of features are extracted as described above and are concatenated with the features extracted from the original data. The values for σ is taken as 3 and 5, where greater values result in more blurriness in the filtered data.

4.2.2 Bayesian Ridge Regression

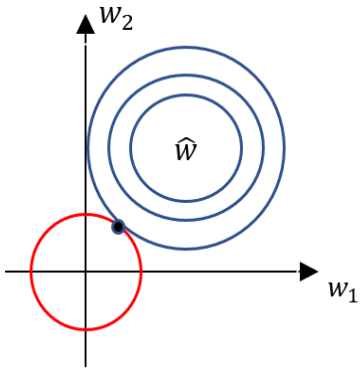


Figure 4.2: Illustration of L2 Regularization.

We start our discussion by explaining Bayes' theorem. *Bayes' theorem* transforms a prior probability into posterior probability depending on the evidence [11]. *Prior* refers to the state of the variable before observing evidence. Posterior is the state of the variable after observing evidence. So the Bayes' theorem can be stated as follows:

$$posterior \propto likelihood \times prior \tag{4.5}$$

We can adopt this concept to determine any model parameter $\mathbf{w}$. Before observing the data, the initial assumption about $\mathbf{w}$ is called the prior probability distribution $P(\mathbf{w})$. The observation of $\mathcal{D}$ is expressed as the likelihood function or the conditional probability $P(\mathcal{D}|\mathbf{w})$. The conditional probability distribution of $\mathbf{w}$ is called the posterior probability, which is $P(\mathbf{w}|\mathcal{D})$ given $\mathcal{D}$. According to Bayes' theorem, the probability of $\mathbf{w}$ after observing $\mathcal{D}$ is as follows:

$$P(\mathbf{w}|\mathcal{D}) = \frac{P(\mathcal{D}|\mathbf{w})P(\mathbf{w})}{P(\mathcal{D})} \tag{4.6}$$

According to frequentist view, the equation for linear regression stands for,

$$\mathbf{y} = \mathbf{w}^T\mathbf{x} + \varepsilon \tag{4.7}$$

where y is the output to be estimated, $\mathbf{x}$ are the features, and ε is the error term. The model parameter $\mathbf{w}$ is found in linear regression by minimizing the residual sum of square (RSS) between the known y values and the estimated ones $\hat{y}$.

$$RSS(\mathbf{w}) = \sum_{i=1}(\mathbf{y_i} - \hat{\mathbf{y}})^2 = \sum_{i=1}(\mathbf{y_i} - \mathbf{w}^T\mathbf{x}_i)^2 \tag{4.8}$$

where N is the total number of observation. By minimizing the above solution, the estimate for $\mathbf{w}$ becomes:

$$\hat{\mathbf{w}} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y} \tag{4.9}$$

In this approach, we only obtain a single estimate for the parameter given the data. In Bayesian regression method, the probability distribution of the parameters is formulated rather than a single point estimate. Here, regression is done in probabilistic ways with explicit priors on the parameters [61]. In other words, the probability distribution of the model parameter is estimated given the features.

The Bayesian regression model starts with same regression equation and assumes that the error ε is independent and normally distributed with zero mean and σ^2 variance. For estimating y out of feature $\mathbf{x}$, the output y is assumed to have a Gaussian distribution with a mean $\mathbf{wx}$ and variance σ^2 with conditioned on $\mathbf{x}$,

$$p(\mathbf{y}|\mathbf{x}, \mathbf{w}, \sigma^2) = \mathcal{N}(\mathbf{y}|\mathbf{wx}, \sigma^2) \tag{4.10}$$

The likelihood of the complete data is as follows:

$$p(\mathbf{y}|\mathbf{x}, \mathbf{w}, \sigma^2) = (2\pi\sigma^2)^{-N/2} exp(-\frac{1}{2\sigma^2}\|\mathbf{y} - \mathbf{x}\mathbf{w}\|^2) \tag{4.11}$$

The feature set extracted from the neighborhood is collinear in nature. This can cause high variance in the estimated parameters in a regression model. In order to reduce the variance of the parameters, a regularization term is added to the loss function that helps in introducing variance to the parameter estimation. If the added penalty term in a regression model is squared magnitude then it is called ridge regression [11]. The RSS with squared error term becomes:

$$RSS(\mathbf{w}) = \sum_{i=1}(\mathbf{y_i} - \mathbf{w}^T\mathbf{x}_i)^2 + \lambda\|\mathbf{w}\|^2{}_2 \tag{4.12}$$

here regularization strength is varied by a hyperparameter λ.

As in Fig. 4.2, the blue contours represent the gradient descent contour plot, where the center point indicates the zone that has the lowest error. The main goal is to reach the center of the contour. But due to adding the square terms to the RSS function, there is a constraint introduced under which the RSS is minimized. The red circular area is the constraint for coefficients in this figure. Intuitively, the coefficients are slightly shifted towards zero to stabilize the importance of the parameters. As a result, without over-fitting, the model can perform a regression analysis. *Overfitting* refers to the situation when the model becomes too complex by incorporating more parameters compared to the number of samples, which leads to error in estimating unknown data samples. In Bayesian ridge regression, the prior for the parameters $\mathbf{w}$ is given by a spherical Gaussian with zero mean:

$$p(\mathbf{w}|\lambda) = \prod_{i=0}^{N} \mathcal{N}(\mathbf{w}|0, \lambda^{-1}) \tag{4.13}$$

here λ is a vector of $N + 1$ hyperparameters. Each parameter is associated with a hyperparameter. So there is a hierarchical prior, where hyperpriors over λ is defined to be gamma distributions, whose parameters are chosen such that they are non-informative (very small values) [92].

4.2.3 Fundamentals of Deep Neural Network

Multi layer perceptron: Neural network is built of small blocks called perceptrons, which takes input values as numbers, multiply them with weights and add biases. The weighted sum is multiplied by an activation function. The weights emphasize on particular nodes and bias values, and thus help in adjusting the activation function for optimal result. The activation function helps in mapping the result in between 0 and 1. In Fig. 4.3, $x_1, x_2,..., x_n$ are the inputs. The parameter w_i is the weight and b is the bias to be added with each input which are usually learnable. Finally, applying activation function f to the total weighted sum of the output y is obtained by,

$$y = f(\sum_{i=1} w_i x_i + b) \tag{4.14}$$

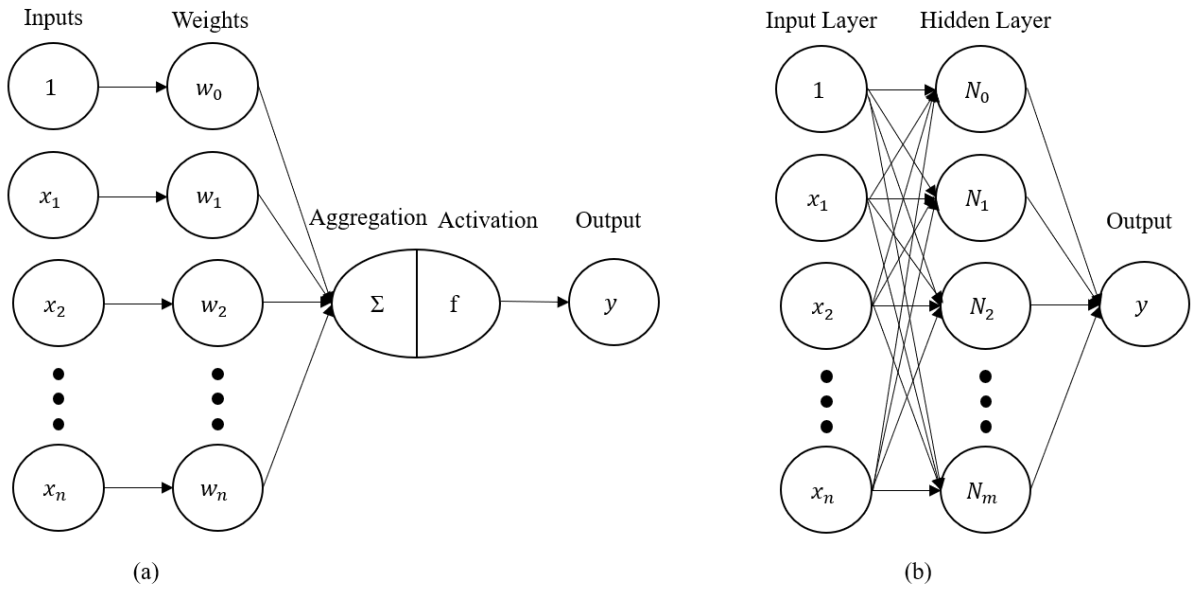


Figure 4.3: Illustration for (a) perceptron. (b) a single layer perceptron or shallow neural network. There are n inputs depending on the features extracted as explained in Section 4.2.4 and there are m neurons in the hidden layer. For this work m is 100.

The stacked perceptron is called *multi layer perceptrons* or *artificial neural networks*. An MLP (Multilayer Perceptron) consists of one input layer, one or more middle layers referred to as hidden layers, and one final layer of perceptrons, known as the output layer. Every layer besides the output layer consists of a bias neuron and is fully linked to the subsequent layer. The multilayer neural network architecture is given in Fig. 4.3.

For each training sample, the inputs are fed to the network and the output of every neuron in each back to back layer is calculated [52]. This step is called the forward pass, which is as same as predicting the output. Then the error between the original output and the estimated output is calculated. Additionally, it calculates the error contribution made by each neuron in the last layer. Then the same error calculation is repeated for every layer by moving backward until the input layer is reached. This switch pass effectively measures the error gradient over all neurons' weights in the system by spreading the gradient in reverse in the system. This backward pass, which minimizes the error, makes the algorithm named backpropagation algorithm. To reduce the error, the algorithm adjusts the weights and biases. So, these are called the adjustable parameters. In a typical deep learning system, there are thousands of weights and biases. Based on the gradient vector, the weights are adjusted to get a minimum error. These weights need to be carefully initialized to get a better result. Different hyperparameters greatly influence the training of a neural network model. Gradient descent, epoch, batch size, activation function are some examples of hyperparameters.

Gradient descent: *Gradient descent* is an iterative algorithm to minimize the error and find the optimal results. It detects the minimum point of the error curve. It can make the under-fitted graph to fit easily with the data. The algorithm takes steps to reach the minimum of the curve in each iteration which is called

learning rate. The learning rate can not be too small or too large. If learning rate is set to a very small value, gradient descent can end up finding the local minima. Whereas, larger steps can speed up the convergence, it can result in long time to reach convergence, as it may bounce back and forth in the convex region [30]. The steps can be decreased with each iteration to converge slow. There are different kinds of gradient descent, for example, stochastic gradient descent, RMSprop, adam optimizer, batch gradient descent.

Epoch and batch size: In the case of training a neural network model, a large dataset is used. The model is trained several times. In each training, the whole dataset is passed forward, and then backpropagation is done. This one set of forward and backward passes is called one *epoch*. The neural network is learned in more than one epoch and in each epoch, the error is minimized using gradient descent. With increasing epoch or the number of times of training, the accuracy increases. The number of epochs can be increased until the accuracy of the validation dataset is decreased. As the whole data can not be feed into the network in one epoch, the data is subdivided into small batches. In one epoch, the model is trained several times using these batches. *Iteration* is the number of times required for one epoch to feed the whole dataset. The required number of iterations is found by dividing data by the batch size.

Activation function: In a neural network, *activation function* decides whether a node will turn on or not in a particular layer. This is also called a transfer function. An activation function maps input and response vector through nonlinear complex functionality. If the activation function f is not included, then the neural network is a linear regression model. With activation function, a complex network can be built based on the pattern of the data. Some of the popular activation functions are rectified linear units(ReLU), Hyperbolic tangent Function (Tanh), and sigmoid. Let x be the input to the activation function. A sigmoid function can be written as $f(x) = 1/(1 + exp(-x))$. It ranges the output between 0 and 1. A hyperbolic tangent function (Tanh) is $f(x) = (1 - exp(-2x))/(1 + exp(-2x))$. *Rectified Linear Units (ReLU)* can be defined as $f(x) = max(0, x)$. ReLU provides zero for all the negative inputs and x for the positive inputs. In spite of being continuous, it is not differentiable at $x = 0$. However, in practice, it works very well and has the advantage of being fast to be computed.

Vanishing gradient problem: *Vanishing gradient* is a problem encountered in gradient based learning of deep neural networks, which makes it hard to train or tune the parameters in the first layers. During back propagation, gradient or derivative is calculated on the output function moving layer by layer from the final layer to the initial one and the gradients are added to the parameters of the layers. A small gradient means less update in parameters and vice versa. If the derivative or gradient becomes zero for some reason, then the weights do not get updated or the model does not learn anything and so accuracy drops. The gradient can become zero for various reasons. For example, the derivative of the activation function being zero causes vanishing gradient. In deeper neural networks, vanishing gradient occurs in the earlier layers.

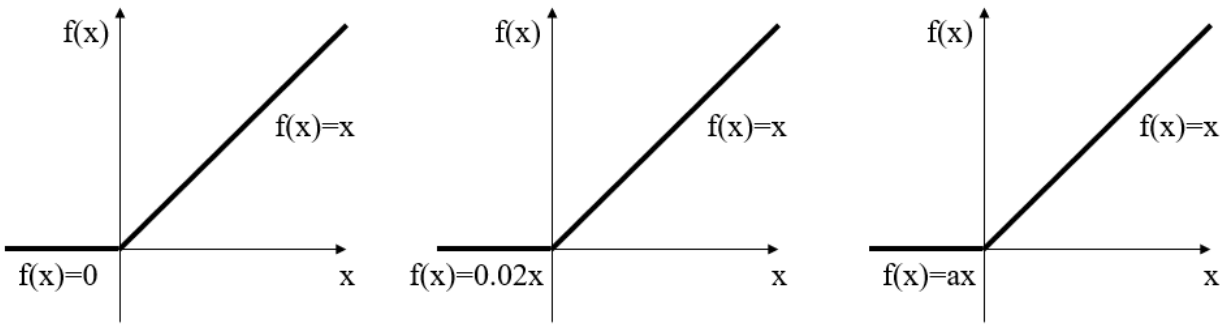


Figure 4.4: Illustration for (left) ReLu. (middle) Leaky ReLU. (right) Parametric ReLU.

Leaky ReLU and Parametric Relu: *Leaky Rectified Linear Units(ReLU)* is a modification of ReLU which does not cancel out all the negative outputs of a layer but multiplies the negative values by a small value. *Parametric ReLU* is same as Leaky ReLU but it learns that small value throughout the training.

$$f(x) = \begin{cases} x & \text{if } x > 0, \\ ax & \text{if } x <= 0 \end{cases} \tag{4.15}$$

here x is the input to the function and a is the learnable parameter.

Batch Normalization: If the input features for any layer in a neural network lie in different range, then the weights and biases of that layer change rapidly and thus the training becomes complex and delayed. Therefore, *batch normalization* scales and adjusts the input features in the hidden layers so that the input features can have a fixed mean and variance. It also reduces covariance shift, which is the change in the distribution of the input features of the hidden layers [39]. Moreover, it solves the problem of vanishing gradient as normalization shrinks the input range.

Residual connection: *Residual connection* connects a layer to its earlier ones. As seen in Fig. 4.5, the residual connection connects the input x to the output of the layer $f(x)$. This is also called skip connection. This skip connection helps in avoiding the gradient being zero by passing it to the earlier layer as it is.

Autoenocder: *Autoencoder* is a type of convolution neural network with a hourglass structure. It has two parts: an encoder and a decoder. The encoder learns a latent space feature representation from its input 2D data which is transformed back to the original data by the decoder as shown in Fig. 4.5. The latent space feature representation usually contains all the high level semantic features of the data.

Convolutional layer: A convolutional layer contains kernels or filters whose parameters are learned during the training procedure. Each filter or kernel convolves with its input image and generates activation map or output. As shown in Fig. 4.6, the kernel or filter slides over the input, and outputs the value of the activation

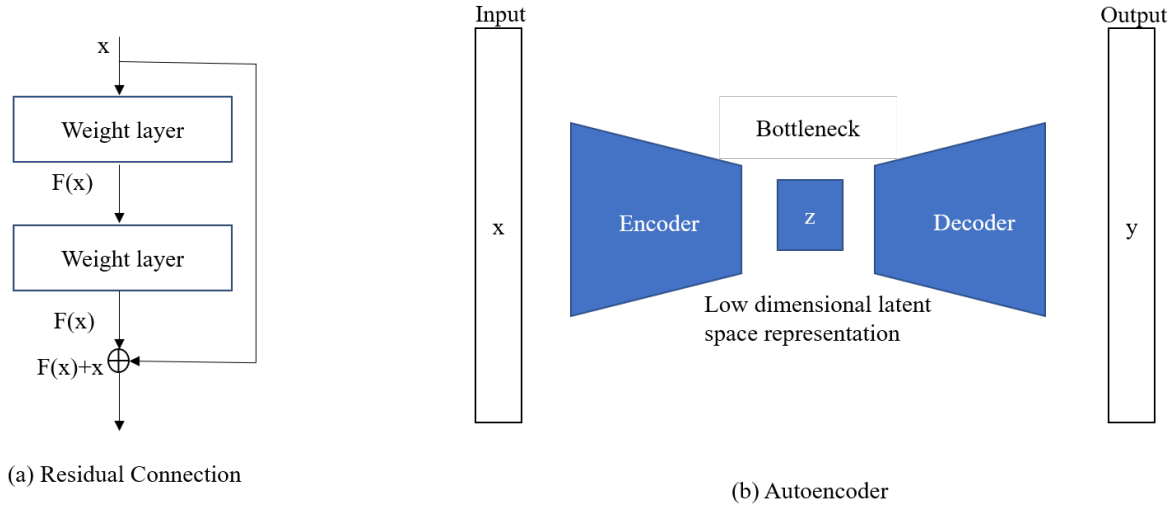


Figure 4.5: Illustration for (a) residual connection. (b) a autoencoder.

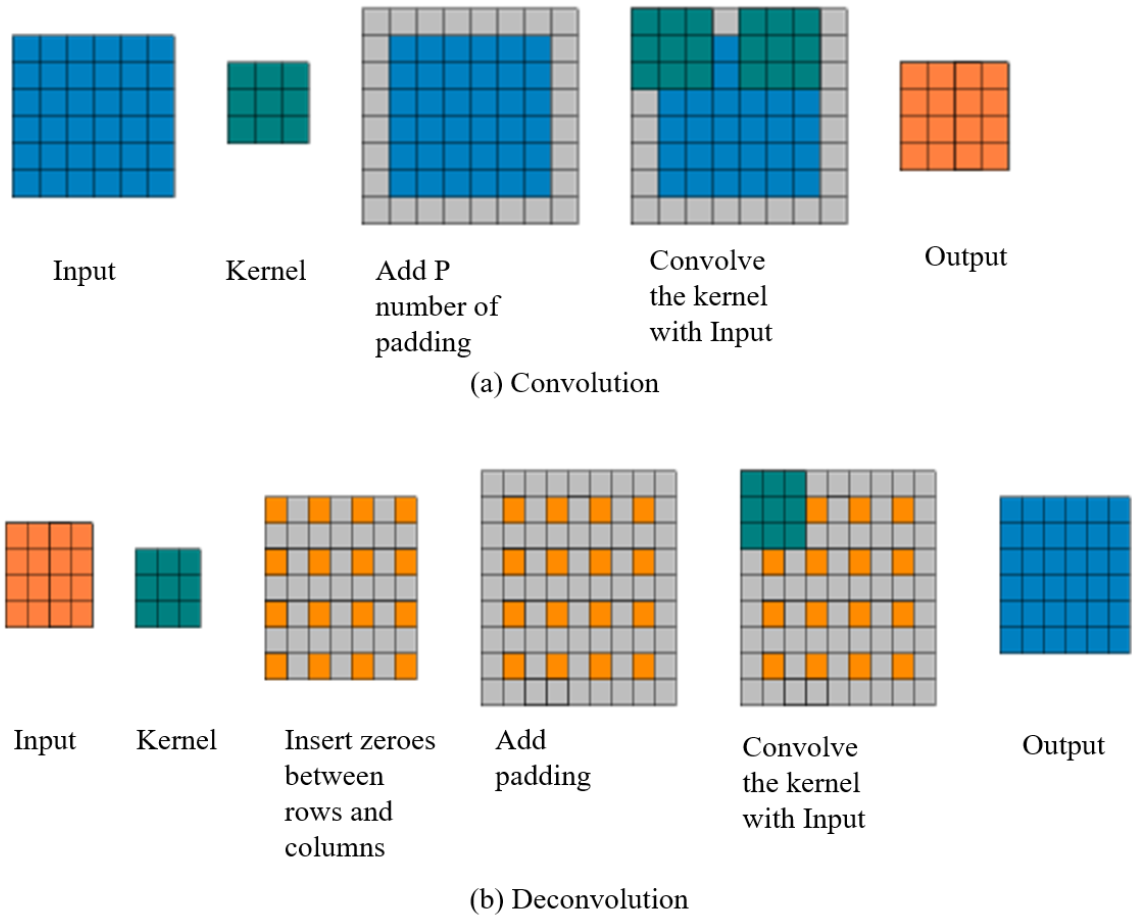


Figure 4.6: Illustration for (a) convolution. (b) deconvolution layers.

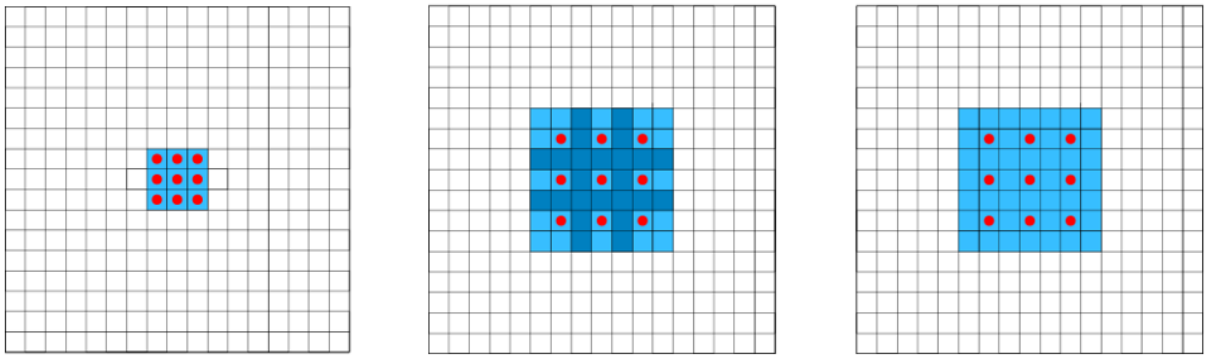


Figure 4.7: Receptive field of dilated convolution layer. (left) Standard convolution layer with 3×3 kernel and 1 dilation rate. The receptive field is 3×3. (middle) Dilated convolution layer with 3×3 kernel and 2 dilation rate. The receptive field is 7×7. (right) Standard convolution layer with 5×5 kernel and 1 dilation rate. The receptive field is 7×7.

.

map. The activation map is formed by repeating the same process for all the elements in the input. The amount of zeros padded around the original input is called padding (P). Stride (S) is the amount by which the kernel is shifted when sliding across the input. Convolutional layer usually reduces the dimension of the output.

Dilated convolution: Contextual information acts as a high level semantic feature, which can promote better understanding of the data [75]. One way to get a better context is using bigger kernels in the convolution layers so that large area gets covered. But this results in more parameters to be trained. This limitation is addressed by *dilated convolution*, which has the advantage of covering bigger area than a regular convolution layer [106]. It is also trained without having any additional learnable parameters. The idea is inspired by Holschneider et al. [36] and Shensa et al [82]. who incorporate holes for wavelet transformation. The idea is to insert holes or zeroes between kernel of convolutional layers to enlarge receptive area, and hence extracting better higher level features. This non-learnable parameter also adds to the computational cost. *Dilation rate* is the number of pixel gap between kernels. As shown in Fig. 4.7, when dilation rate is 1, it is basic convolution with kernel size 3×3. But if the dilation rate is 2, the receptive field increases. In order to achieve the same receptive area, the kernel size has to be 5×5 in regular convolution layer, which will require 25 parameters to learn. The common convolution layer has a linear correlation with receptive field which is $(2i-1) \times (2i-1)$. A dilated convolution has an exponential correlation, e.g. $(2^{i+2}-1) \times (2^{i+2}-1)$, while the number of parameters grow linearly [106].

Deconvolutional layer: *Deconvolutional layer* generates activation map that has higher dimension than the input. It is used for upsampling. Deconvolutional layer works the same way as convolutional layer by convolving the kernel with the input. It also has padding and strides. The deconvolutional layer inserts zeros

between the rows and columns, as shown in Fig. 4.6.

Subpixel convolution layer: *Subpixel convolution layer* is used for upscaling that uses convolutional approach along with a phase shift. Using phase shift, the layer shuffles the input of dimension $H \times W \times C.r^2$ to have shape of $rH \times rW \times rC$, where r is a fixed integer.

4.2.4 Shallow Neural Network

For this work, a single layer artificial neural network is applied on different input features based on 3 and 5 kernels as explained in Section 4.2.1. The architecture is shown in Fig. 4.3. There are 100 nodes in the layer. The model optimizes squared loss using stochastic gradient descent algorithm.

4.3 Generative Adversarial Network (GAN)

The problem of recovering deleted data can also be thought of as the problem of generating the data. Hence we attempted to model the distribution of the deleted data through generative models. Here the model is trained to learn the data distribution, and the hope is that the output generated based on the learned data distribution will closely match the original data. Among the generative models, generative adversarial network (GAN), proposed by Goodfellow [31], has had tremendous success over the other generative models (Variational Autoencoders [49, 77], Naive Bayes Model [59]). Such an approach has been widely used for image generation [40], music generation [105], drug design [46]. In the following, we first describe GAN architecture and then review its training process.

4.3.1 Vanilla GAN

The vanilla GAN is built of two networks: a generator G and a discriminator D. The input to G is a random variable having a prior distribution either Gaussian or uniform distribution. The task of the generator is to learn the distribution of the true samples x_t and generate fake samples x_g which are very similar to the true samples. Let the mapping function by generator be defined as $x_g = G(z, \theta_g)$, where z is the input random variable, and θ_g is the parameters of generator model. The discriminator D, being a binary classifier, differentiates between the fake generated sample and true samples. So the input to D is either x_t or x_g and the output is zero or one, indicating true and fake samples, respectively.

The optimization of this model follows the minimax algorithm [21], where the two models compete with each other to become better at generating and discriminating. The aim of discriminator is to get better at discriminating between the true samples and generated samples, whereas the generator tries to confuse the discriminator by generating samples very similar to true samples. In other words, the loss function is minimized by G but maximized by D similar to a two player minimax algorithm. The loss function is as follows:

$$\min_G \max_D L(G, D) = \mathbb{E}_{x_t \sim P(x_t)}(log(D_{x_t}) + \mathbb{E}_{x_g \sim P(x_g)} log(1 - D(G(z)))) \tag{4.16}$$

where $D(x_t)$ is the probability estimate for real data x_t by discriminator, $\mathbb{E}_{x_t \sim P(x_t)}$ is the expected value over all the real data, $G(z)$ is the generator's output given random variable z, $D(G(z))$ is the discriminator's probability estimate for generator's output being real, $\mathbb{E}_{x_g \sim P(x_g)}$ is the expected value over all random inputs to the generator. The term $\min_G$ refers to minimization of the function with respect to G. The generator minimizes the log of the inverse probability predicted by discriminator for its generated data. This helps in generating good samples that have low probability of being fake. The term $\max_D$ refers to maximization of the function with respect to D. The discriminator maximizes both the log probability of real images and the log of the inverse probability for fake images.

The gradient is backpropagated to update the generator parameters θ_G such that the generator can generate data that can fool discriminator. Minimizing the above function is equivalent to minimizing the Jensen-Shanon (JS) divergence [31]. The final outcome of this training process is the samples x_g, which explicitly or implicitly have a probability distribution like true data distribution $P(x_t)$. On the contrary, discriminator will show 50% probabilities of identifying true and generated data which means that generated data will be indistinguishable from the true ones.

4.3.2 Super-resolution Generative Adversarial Network (SRGAN)

For reconstruction of the deleted data points, we make use of Super-resolution Generative Adversarial Network (SRGAN) [53]. SRGAN upscales low-resolution images into high-resolution mages. A reduced data of $\frac{m}{c} \times \frac{n}{c}$, where $c > 0$, dimension is taken as input to the model and the model estimates the $m \times n$ shaped data. Here c is the compression ratio. Given a matrix data, we first remove 75% of the data by grid deletion, and then (from the remaining data samples) create a contour plot, i.e., every pixel of the contour plot image represents a data sample. This is issued as the low resolution image input, as shown Fig. 4.8.

The SRGAN architecture consists of generator and discriminator models. The generator is trained to learn the mapping between deleted data and entire original data, whereas the discriminator is trained to differentiate between the data generated by generator network and entire original data. The architecture of the network is shown Fig. 4.8. The architecture endorses the Residual Network (ResNet) [35]. In a regular feed forward network, each layer feeds into the next layer. But ResNet introduces skip connection in the network, which connects two layers, that are distant more than one layer from each other. This can mitigate the problem of having vanishing gradient in very deep networks. It consists of one pre-residual layers, 16 residual blocks, and finally, one subpixel convolutional layers. The residual blocks are identical in structure each having two convolutional layer followed by batch normalization layer and Parametric ReLU as activation layer. The subpixel convolutional layer upsamples the data by c factor, which is the compression ratio. The architecture of the discriminators has eight convolutional layers with batch normalization layers, and Leaky ReLU as activation layer. Then there are two dense layers and a sigmoid activation function. More details

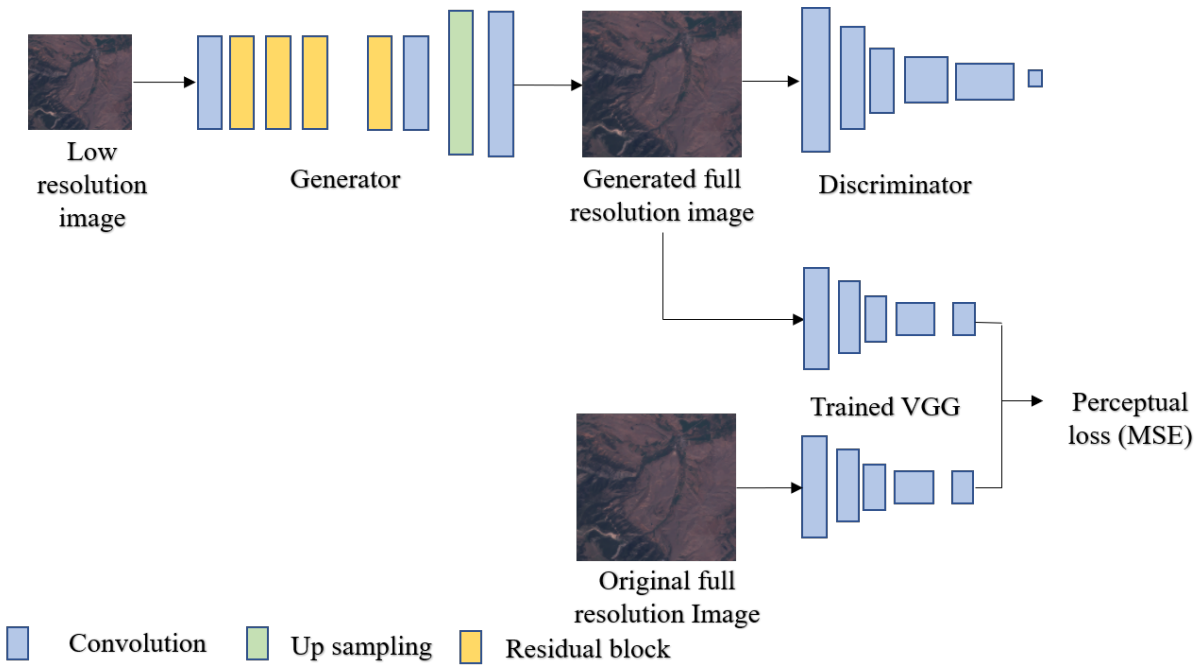


Figure 4.8: SRGAN network architecture.

are provided in the Appendix (Table A.1 and Table A.3). SRGAN optimizes a specially designed perceptual loss function [84], which is capable of maintaining finer details such as texture, content in the reconstructed full data.

Generator loss is defined as follows:

$$L_G = L_{content} + \lambda L_{Adversarial} \tag{4.17}$$

here λ is a scaling factor set to 10^{-3} in this work based on prior research [53]. The content loss includes not only the mean square error (MSE) loss at pixel level, but also the mean square error at feature level. For the later part, feature maps are generated from the 19 layer pretrained VGG network [84]. VGG means Visual Geometry group, which developed a convolutional neural network, which achieved very high accuracy on ImageNet dataset. The idea of using VGG network is that the trained VGG network is capable of understanding the features in the images. So by comparing the latent space feature representation for both high and low resolution image, the model can learn the realistic formation of high resolution image. The equations for pixel level loss L_{mse} and feature level loss L_{VGG} are given below:

$$L_{mse} = \frac{1}{c^2 mn} \sum_{i=1}^{cm} \sum_{j=1}^{cn} (I_{i,j}^F - G(I_{i,j}^C))^2 \tag{4.18}$$

$$L_{VGG} = \frac{1}{m_{x,y} n_{x,y}} \sum_{i=1}^{cm_{x,y}} \sum_{j=1}^{cn_{x,y}} (\phi_{x,y}(I_{i,j}^F) - \phi_{x,y}(G(I_{i,j}^C)))^2 \tag{4.19}$$

where $I_{i,j}^C$ is the reduced data and $I_{i,j}^F$ is the full data, m and n denote the width and height of the reduced data, respectively. In the VGG network, before yth convolution and before xth max pooling layer,

$\phi_{x,y}$ denotes the feature map obtained with dimension $m_{x,y}$ and $n_{x,y}$. Finally, G denotes the reconstructed data by the generator model.

The adversarial loss is as follows:

$$L_{adversarial} = \sum_{n=1}^{N} -\log(D(G(I^C)) \tag{4.20}$$

here I^C is reduced data, N is the number samples in each batch. D and G denotes the parameters of their networks. The above equation is minimized by generator by generating full data that are very close to original full data. Comparing this loss function with the one from vanilla GAN, it is seen that instead of $[1 - \log(D(G(M^C))]$, generator is aimed to minimize $-\log(D(G(M^C))$. This is to achieve better gradients and more stable training [53]. In the beginning of the training, the generated data is usually not close to the true one. A two step training for Super resolution GAN model is applied that ensures a converging loss. Adam optimizer has been used in both of the steps. In the first step, the generator network is trained with a MSE based loss only. The trained weights are taken as initialization for the SRGAN network and further trained with both content and adversarial loss.

4.3.3 Temporal Inpainting

Image inpainting is a task of filling unknown pixels. While inpainting, traditional methods often fail to create sharp edges or generate enough details, and end up delivering a blurry result. As a result, a two-step network is proposed by Jiahui et al. [107], where the first network reconstructs the unknown pixels coarsely and the second network refines the coarse result from the previous network as shown in Fig. 4.9. There are two discriminators called global and local critics. Global critics looks at the global image and local critic looks at only the missing locations and assigns how much accurately the model is generating the global and local image pixels, respectively. As Jiahui et al. uses the work of Iizuka et al. [38], each of two networks (coarse and fine) has architecture as almost same as that of Iizuka et al.'s. Moreover, this architecture implements a contextual attention layer, which is capable of successfully borrowing similar and available texture or color from the surrounding. Motivated by all these effective strategies, we leverage their network as a base for our reconstruction model.

Jiahui et al.'s model consists of two networks: coarse and refinement network. The input to the coarse network is input with mask as shown in Fig. 4.9. This network consists of a convolution layers, dilated convolution layers, up sampling layers and finally more convolution layers. The output is a blurry construction of pixels in the unknown pixel positions. This coarse result is taken as the input to the refinement network which is passed through the two pipelines each having contextual layers and dilated convolutional layers respectively. The coarse network provides a somewhat blurry structure in the unknown position which is further refined. If instead of blurry reconstruction, the shape information is given, then through refinement it will be possible to get cleaner shapes and reconstruction. This concept has paved the way to modify their network in such a way that a better reconstruction can be achieved.

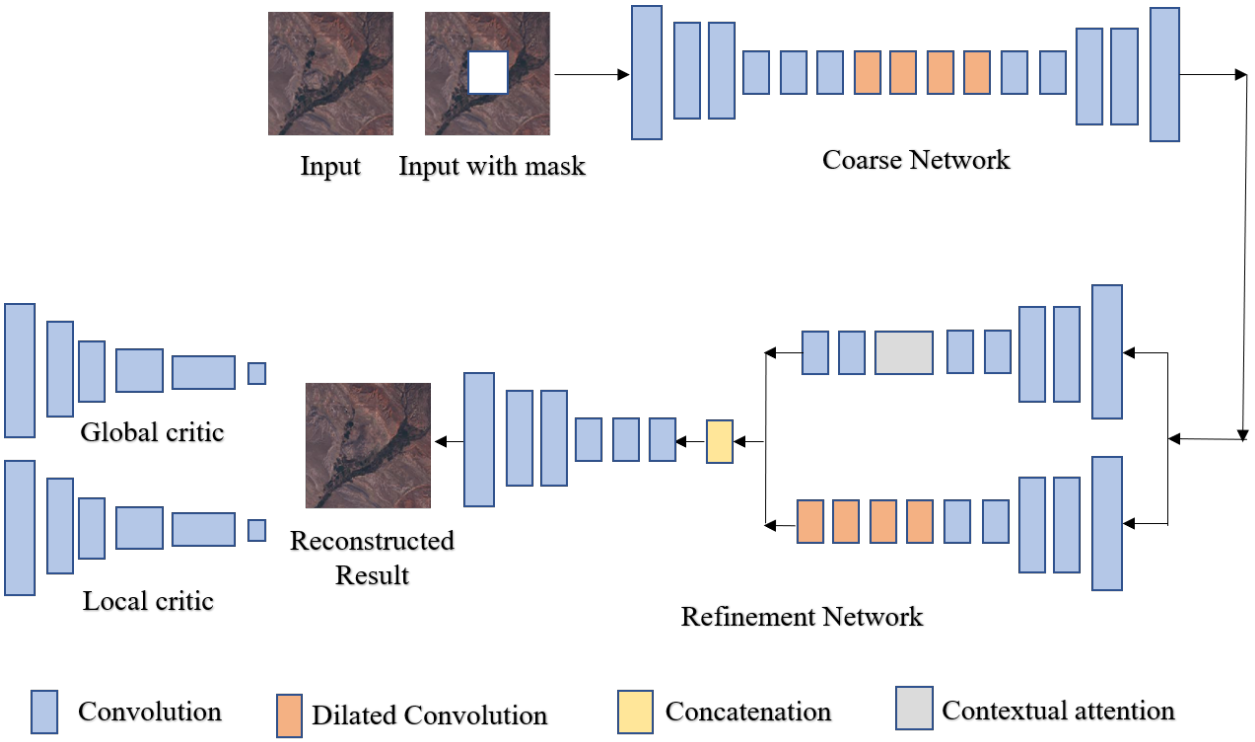


Figure 4.9: Inpainting network by Jiahui et al. [107].

Our Modification: The original inpainting pipeline fails to reconstruct complex and detailed shape as it depends on the reconstruction of the coarse result from the first network. If the information of the shape is provided better in the coarse result, then the second network will be better at producing a polished reconstruction. Since the geospatial datasets and satellite imagery often have some auxiliary data like temporal or spectral, we attempted to fill in the deleted points of the input data by any available data of the same place from any other timestamp (i.e., temporal data) or some data interpolation technique. Therefore, instead of an image multiplied with mask as input, we feed the image inpainted by temporal data or interpolation as input to the refinement network. This helps greatly in converging and also building a better reconstruction. The architecture is shown in Fig. 4.10.

Overview of the architecture: The full architecture consists of one autoencoder network as same as the 'refinement network' as shown in Fig. 4.9. The input to the network is the reduced data filled with temporal or interpolated data and mask. Mask is composed of zeros and ones, where zero denotes the deleted sample points and one denotes the non deleted ones. There are two encoders connected in parallel. One encoder is capable of extracting feature from the input image through dilated convolution layers. The second layer brings features from the available pixels. The two latent space representation is concatenated and fed into the decoder network. The result obtained from this network is the reconstructed image, which is scored by global and local critics, and helps in training the network. The layers of the models are of four types: convolution, dilated convolution, concatenation, and contextual attention.

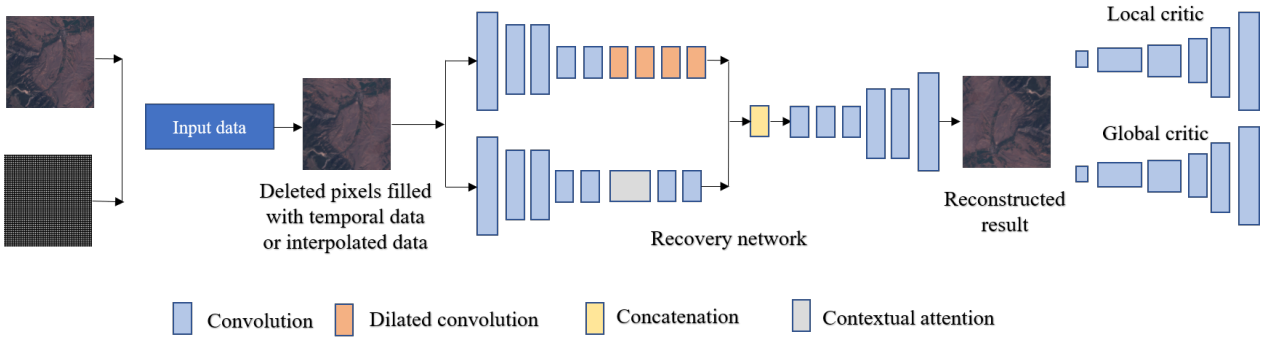


Figure 4.10: Recovery network architecture.

In the contextual attention module, 3×3 patches are extracted from the whole image and are reshaped as convolutional filters as shown in Fig. 4.11. They are convoluted with each unknown pixels for measuring cosine similarity which is computed as follows:

$$s_{x,y,x',y'} = \langle \frac{k_{x,y}}{\|k_{x,y}\|}, \frac{u_{x',y'}}{\|u_{x',y'}\|} \rangle \tag{4.21}$$

where $s_{x,y,x',y'}$ is the similarity metric between known pixel $k_{x,y}$ and unknown pixel $u_{x',y'}$. The cosine similarity for the highest corresponding pixel is directly copied from the surrounding. This helps in retaining the color of the spatial image. Although the temporal data file of the same place may contain the shapes, which are almost similar to the deleted file, there might still be visual differences between them, particularly in color (e.g., satellite images in winter and summer). Contextual attention helps solve this issue by bringing better color information. The similarity matrix is scaled using a softmax function along each channel:

$$softmax(\mathbf{p}) = \frac{exp(-p_i)}{\sum_{i=1}^{N} exp(p_i)}, \text{for } i = 1, ..., N \text{ and } \mathbf{p} = (p_1, ..., p_N) \tag{4.22}$$

where $\mathbf{p}$ is the input vector. The entire output of the function adds up to one so that they can be considered as probabilities. The output of the softmax layer is scaled and weighed to obtain the patches with higher attention score. The extracted patches with higher attention score are used for deconvolution. The contextual layer details is given in Appendix (Table A.4). Intuitively, for an unknown pixel position, this module learns where to bring the pixel information from among the known pixels.

Global and Local Critics: The GAN network is based on the concept of Wasserstein distance. So it follows the concept of WGAN-GP [5]. Hence, the discriminators here are called critics as they no longer just differentiate between true and generated image, rather provide a score or measure of how close the generated data is to the true data. The global critic takes the whole generated data which has a shape of 256×256 as input and local critic takes only the reconstructed data as input which has a shape of 128×128. WGAN computes the Wasserstein distance $W(P_t, P_g)$ between true samples P_t and generated samples P_g. Using the

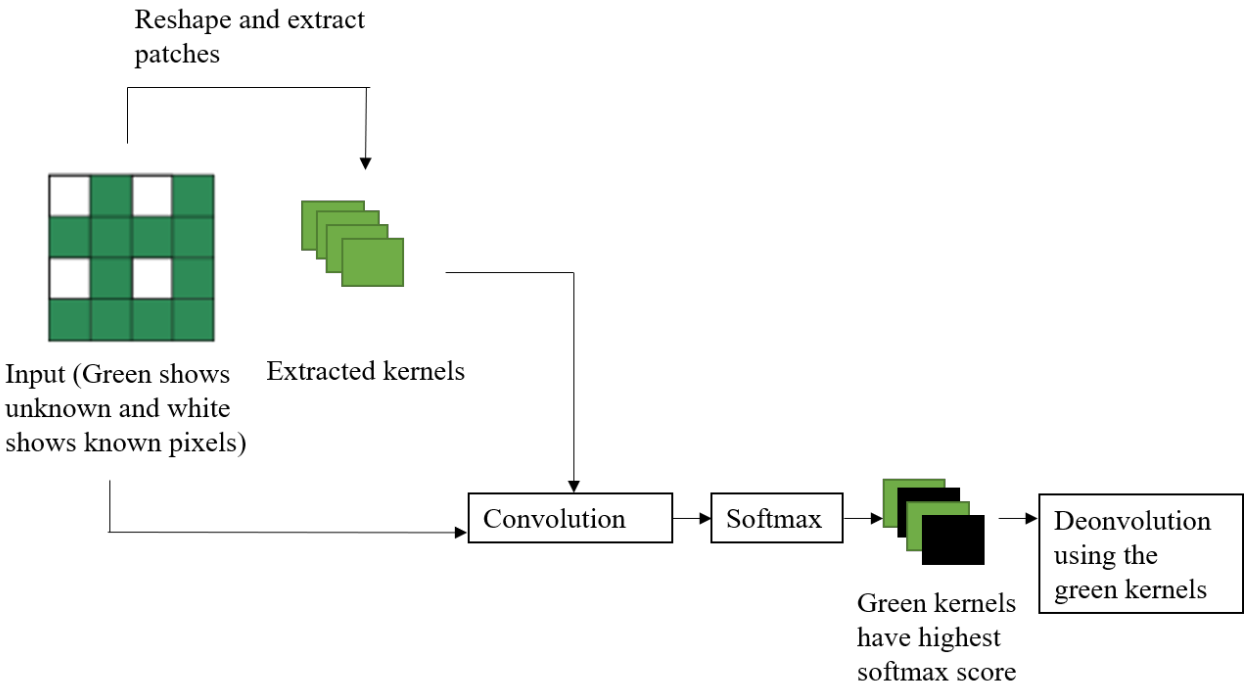


Figure 4.11: Illustration for contextual attention layer.

Kantorovich-Rubinstein duality, the loss function is as follows:

$$\min_{G} \max_{D \in \mathbb{D}} L(G, D) = \mathbb{E}_{x_t \sim P(x_t)}(log(D_x) + \mathbb{E}_{x_g \sim P(x_g)} D(G(z))) \tag{4.23}$$

where $\mathbb{D}$ is the set of 1-Lipschitz functions and P_g is the distribution of the generated data for $x_g = G(x_t)$. Minimizing this function with respect to generator results is equivalent to minimizing wasserstein distance between P_t and P_g [32].

4.4 Summary

In this section, we have described the recovery methods. Interpolation methods, Bayesian ridge regression, and a shallow neural network are chosen from traditional machine learning methods. Among them, the shallow neural network worked better on the features extracted from the neighbourhood. Finally, two types of GAN model are taken into consideration, which are SRGAN and GAN based inpainting. SRGAN takes reduced data as input and generates full data. We observed that directly adopting traditional inpainting method produces blurry output as it attempts to collect perceptual information from another pixels which are not consecutive due to grid deletion. We have modified the traditional inpainting network, where the input to inpainting network is the masked image initially filled with temporal data or data interpolation which helps the model fill in the unknown positions with better color and shapes of objects. The performance of these models are explained in Chapter 5.

Chapter 5

Experimental setups & Results

In this chapter, we first describe the datasets used for the experiments. We then describe the dataset source, acquisition process, and preprocessing. Afterward, we review the modes that we use for running the results, the evaluation metrics. Finally, we discuss the experimental results.

5.1 Experimental Setups

5.1.1 Dataset Description

We performed experiments on a weather dataset [86] and a satellite dataset created from USGS satellite data archive [1].

Weather Dataset: The *Weather Research & Forecast (WRF)* dataset ranges from the year 2008 till 2015 (8 years) [55]. Each data file consists of hourly data for a single day over 699 latitude and 639 longitude points of western Canada. Each day is stored as a NetCDF file (approx. 1.3 GB) containing 10 million (10,719,864) samples, each sample having around 36 weather parameters.

For our experiment, we have picked data of soil moisture and albedo parameters for January of the years 2013 to 2015. We have chosen the variables, soil moisture, and albedo, as they are often used in meteorological research. *Soil moisture* is a variable that measures the water held in land surface [73]. *Albedo* measures the amount of radiation reflected by the earth's surface [71]. First, CSV files are extracted from the netCDF files. For temporal inpainting, we make pairs of these data files. Each temporal pair is selected randomly among the available files. The maximum temporal distance between the pairs is 3 years with no file repeated. Later, patches are cropped out from them and the temporal order is maintained. We used a perceptual color map [68] for generating the contour plots from the weather datasets. Since both SRGAN and image inpainting based architectures consider perceptual loss function while reconstructing an image, the choice for the color map is an important determining factor for the reconstruction performances. The workflow is shown in Fig. 5.1.

Satellite Dataset: We used 10 images captured by Landsat Thematic Mapper satellite which are of $1720 \times 2040 \times 6$ dimensions [1]. We also collected 10 more temporal images, of the same area, which has a

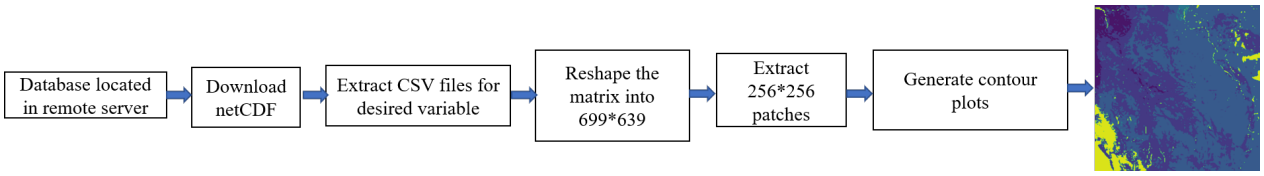


Figure 5.1: Workflow for weather data extraction

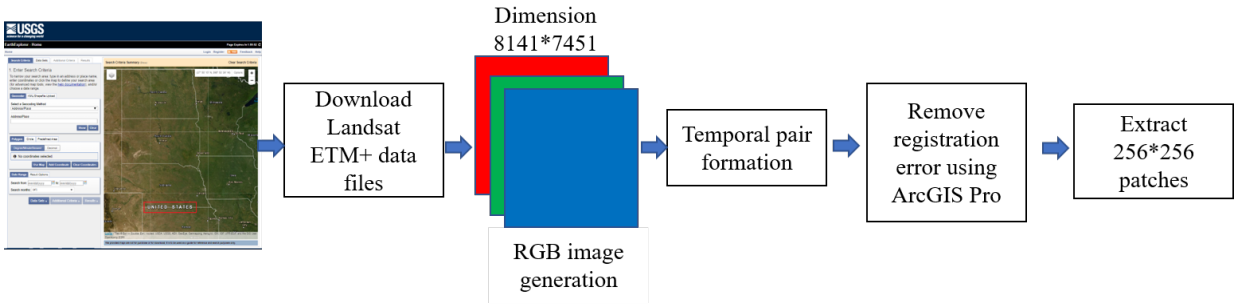


Figure 5.2: Workflow for satellite data extraction.

temporal difference of a maximum of 3 years. For our experiment, we consider red, green, and blue channels only. The temporal pairs have a registration error, which is corrected using ArcGIS Pro software. Then, we cropped 256×256 tiles out of these big images. The timestamp difference of temporal images is not more than 3 years. The steps are illustrated in Fig. 5.2.

5.1.2 Line of work

Machine Learning: The traditional machine learning algorithms are applied to the dataset obtained after a grid deletion. So the first step is to apply grid deletion to input matrix data as shown in Fig. 5.3. Later, features are extracted for each of the deleted samples. In order to validate the model training, the dataset is split into train and test set using a commonly used 80-20 ratio. Then these features are fed into the machine learning models, i.e., Bayesian ridge regression and shallow neural network. We have considered Bayesian ridge regression as the baseline approach. The results are analyzed in two ways. First, the MSE, MAE, $R^2 score$ are calculated between the original matrix and the reconstructed matrix. Second, contour plots and images are generated from weather and satellite matrix data, respectively. The generated images from reconstructed matrix data are compared with that from original matrix data using MSE, SSIM, and PSNR. The naming convention for this pipeline is modelName_Kx_σy. BR stands for Bayesian Regression, SNN stands for shallow neural network, k stands for the number of kernels and σ is the standard deviation of Gaussian filter, which was applied to the image for feature extraction.

SRGAN: The first step is to reduce the data dimension following the data deletion strategies namely, checkerboard, grid, and variance based deletion which will result in 256×128, 128×128, and 128×128 shaped data, respectively. From the reduced data, contour plots for weather and color images for satellite

data are created. Similar to the machine learning model, a train-test split is done on the image data sets. The training set is fed to the generator model and then the model is trained with only MSE loss. After the training of the generator model, the weights are used as initialization, and the whole GAN model is trained. Finally, the test image data sets are used to test the model performance by using the same image evaluation metrics. The output of this model is full resolution data which has a shape of 256. The pipeline is explained in Fig. 5.3.

Inpainting: For the inpainting model, it is necessary to have a mask. The mask is of the same shape as the whole data, containing zeros and ones where zero represents missing pixels and one represents available pixels. Both input and output shape is always 256×256. The training of the model is done in one stage. we examined two approaches to generate a better intermediate coarse image in the image inpainting model. One that fills the missing data points from temporal data, i.e., another image of the same location from a different timestamp. We trained and tested this model with images filled in by temporal data and the notation for this approach is Temp_Inpainting. Since temporal data may not always be available, we also tested the models with interpolated data, using a simple interpolation (Navier–Stokes interpolation [10]) and the notation for this pipeline is Temp_Inpainting-Interp. We trained another model with inputs interpolated in the unknown positions. We tested this model with interpolated values in the unknown positions. The notation for this approach is Interp_Inpainting.

Variance based deletion: For Variance based deletion, all the reconstruction steps are the same. After applying grid based deletion and then the variance based deletion, the unknown positions due to variance based are filled using interpolation.

Technical details: We have trained our deep learning models[1] with TensorFlow 1.14, CUDNN version 10.0, CUDA version 10.0 on a machine of Tesla V100 GPU. The SRGAN and inpainting network took 48 hours and 72 hours, respectively, for training.

5.1.3 Evaluation Metrics

For quantitative comparison of the reconstructed images, we consider the commonly used evaluation metrics such as Structural Similarity Index (SSIM), Signal to Noise Ratio (PSNR), and Mean Squared Error (MSE) [37]. For MAE and MSE, lower values are expected, but for the coefficient of determination, a higher value is expected. The reconstructed matrices were evaluated by MSE, Correlation of coefficient, and finally, mean absolute error (MAE). For SSIM and PSNR, larger values are expected but, for MSE, a lower value is expected. The definition of the evaluation metrics are given as follows:

[1]The codes are available at https://github.com/jat923/Data-Reduction-and-Deep-Learning-Based-Recovery-for-Geospatial-Visualization-and-Satellite-Imagery.git

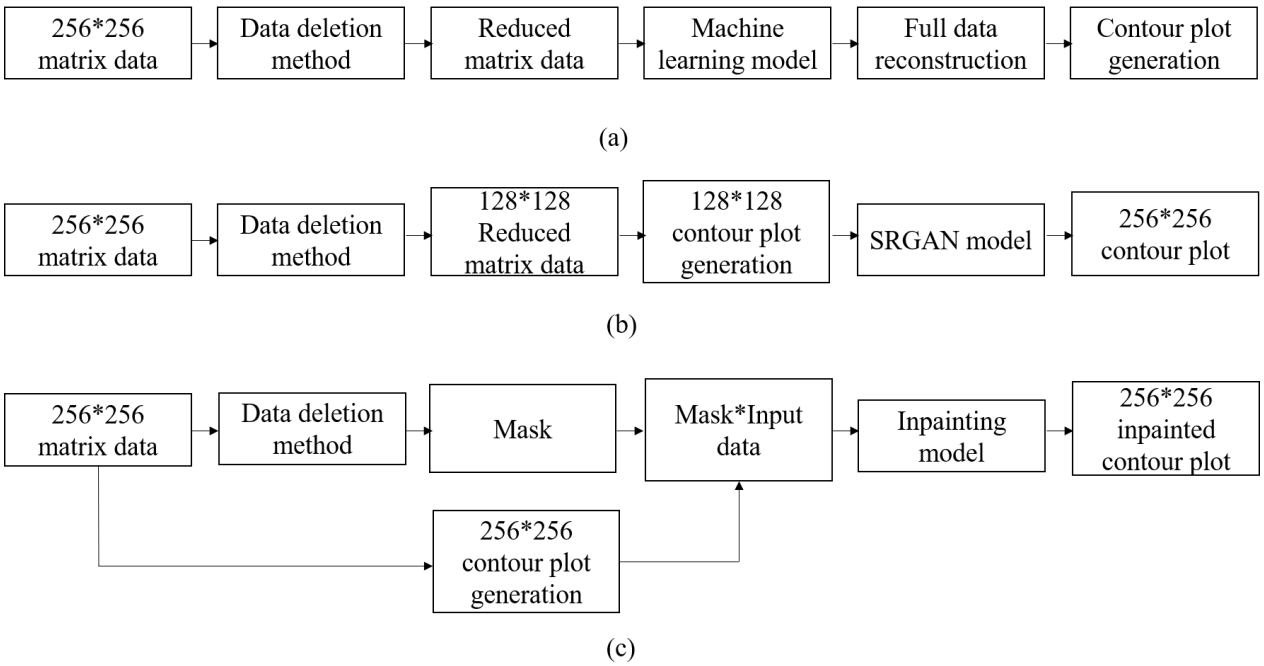


Figure 5.3: Illustration of the steps for running experiments for (a) machine learning, (b) SRGAN, and (c) Inpainting model.

Mean Squared Error (MSE): The most simple yet important measure is mean squared error. For images, *MSE* is defined for a pixel to pixel squared error. Likewise, for matrix data, MSE is the squared error between each original and reconstructed or predicted data point. The lower the value, the better is the prediction. Zero means no error between original and reconstructed or predicted data point.

$$MSE = \frac{1}{WH}\sum_{i=1}^{W}\sum_{j=1}^{H}(y_{ij} - x_{ij})^2 \tag{5.1}$$

Here, y is the original data and x is the recovered data.

Peak signal to noise ratio (PSNR): PSNR is denoted by,

$$PSNR = 10 \cdot \log_{10} \frac{v^2}{MSE} \tag{5.2}$$

Here, v is the maximum value of the image pixels. A higher PSNR value indicates better picture quality. For image analysis, MSE provides a measure of error or noise. By contrast, PSNR is a denoised form of measure. However, both of the metrics fail in terms of assessing blurry results that lack details [53].

Structural similarity index (SSIM): SSIM quality measure is built as a combination of luminance, contrast and structure comparison [114].

$$SSIM(x, y) = l(x, y)^{\alpha}, c(x, y)^{\beta}, s(x, y)^{\gamma}, \tag{5.3}$$

Table 5.1: Matrix reconstruction performance comparison for different strategies (green indicates a better performance when comparing the models).

Model	Soil moisture			Albedo			Satellite		
	r^2	MSE	MAE	r^2	MSE	MAE	r^2	MSE	MAE
BR_k3	0.6746	0.018	0.047	0.5635	0.0141	0.0548	0.8743	0.0007	0.0157
BR_k5	0.6655	0.0185	0.0496	0.5754	0.014	0.0559	0.8423	0.0009	0.018
SNN_k3	0.6728	0.0182	0.0459	0.594	0.0129	0.0499	0.8806	0.0007	0.0152
SNN_k5	0.6746	0.018	0.0471	0.5901	0.0133	0.0533	0.8463	0.0009	0.0176
BR_k3_σ3	0.6757	0.0179	0.0503	0.5976	0.0131	0.0531	0.8636	0.00084	0.01654
BR_k3_σ5	0.6699	0.0183	0.0512	0.5925	0.0132	0.0535	0.8616	0.0008	0.01666
BR_k5_σ3	0.6686	0.0183	0.0493	0.5893	0.0134	0.0528	0.8495	0.0009	0.01762
BR_k5_σ5	0.6686	0.0183	0.0493	0.5893	0.0134	0.0528	0.8495	0.0009	0.01762
SNN_k3_σ3	0.6832	0.0174	0.0489	0.5929	0.0133	0.0529	0.8804	0.0007	0.0152
SNN_k3_σ5	0.6802	0.0176	0.0491	0.5929	0.0132	0.0521	0.8675	0.0007	0.01607
SNN_k5_σ3	0.6794	0.0177	0.0457	0.6062	0.0126	0.0522	0.84091	0.0009	0.01818
SNN_k5_σ5	0.6798	0.0177	0.0461	0.6031	0.0127	0.0515	0.8552	0.00088	0.017
Multiquadrics	0.7599	0.0131	0.0313	0.6888	0.0093	0.0334	0.8642	0.0012	0.0246
Shepard's IDW	0.6681	0.0183	0.0429	0.6544	0.0131	0.0476	0.8838	0.0007	0.015

where

$$l(x,y) = \frac{2\mu_x mu_y + C_1}{{\mu_x}^2{\mu_y}^2 + C_1} \tag{5.4}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{{\sigma_x}^2 {sigma_y}^2 + C_2} \tag{5.5}$$

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x sigma_y + C_3} \tag{5.6}$$

Here, μ_x,μ_x,σ_x,σ_y and σ_{xy} are mean, standard deviation and covariance of two images. The parameters α, β, and γ are greater than zero in value which assign importance to each of the components. The parameters C_1, C_2, and C_3 are constants. SSIM is capable in capturing local luminance and structure. SSIM ranges from 0 to 1, where 1 means perfect match between original and predicted image.

Mean absolute Error(MAE): *Mean absolute error* is a regression loss which computes mean absolute error between the original and predicted data. This is also l_1 norm loss. Lower values are expected for MAE. The equation for this metric is as follows:

$$MAE = \frac{1}{WH}\sum_{i=1}^{W}\sum_{j=1}^{H}|x_{ij} - y_{ij}| \tag{5.7}$$

Table 5.2: Image reconstruction performance comparison for different strategies (green indicates a better performance when comparing the models).

Model	Soil Moisture			Albedo			Satellite		
	SSIM	PSNR	MSE	SSIM	PSNR	MSE	SSIM	PSNR	MSE
BR_k3	0.936	20.7321	0.0092	0.8834	22.6564	0.0165	0.9258	31.6418	0.0007
BR_k5	0.9279	20.2007	0.0109	0.9095	22.9651	0.0057	0.9028	30.5271	0.0009
SNN_k3	0.9013	17.0922	0.0307	0.9167	19.6284	0.0047	0.9287	31.9299	0.0007
SNN_k5	0.9315	20.3008	0.0107	0.9152	22.6419	0.0049	0.907	30.7819	0.0009
BR_k3_σ3	0.9357	22.2356	0.0067	0.9152	25.7043	0.0048	0.918	31.1368	0.0008
BR_k3_σ5	0.9342	22.1387	0.0069	0.9143	25.6998	0.0048	0.918	31.13685	0.00086
BR_k5_σ3	0.9352	22.1735	0.0068	0.9137	25.4581	0.0051	0.907	30.765	0.0009
BR_k5_σ5	0.9352	22.1735	0.0068	0.9137	25.4581	0.0051	0.907	30.765	0.907
SNN_k3_σ3	0.9401	22.2931	0.0068	0.8803	22.5949	0.0222	0.9289	31.8697	0.0007
SNN_k3_σ5	0.9396	22.1966	0.0069	0.9081	24.6087	0.0086	0.9252	31.5693	0.00077
SNN_k5_σ3	0.9385	22.0968	0.0079	0.9177	25.8642	0.0047	0.9051	30.64278	0.00096
SNN_k5_σ5	0.9372	21.7887	0.0092	0.9194	26.0814	0.0042	0.91032	31.0012	0.0008
Multiquadrics	0.912	18.4844	0.0181	0.8894	22.8652	0.0149	0.8894	22.8652	0.0149
Shepard's IDW	0.9373	22.6212	0.0061	0.9188	26.9014	0.0046	0.8783	28.3589	0.0014
SRGAN	0.8943	25.6858	0.0029	0.8242	24.324	0.00433	0.7952	20.1861	0.0208
Temp_Inpainting	0.9948	41.5267	0.00007	0.977	34.5616	0.0004	0.981	39.5178	0.0001
Temp_Inpainting-Interp	0.9949	41.7165	0.00007	0.9801	35.1465	0.0004	0.9909	41.9608	0.000008
Interp_Inpainting	0.9931	40.0353	0.0001	0.9801	35.1465	0.0003	0.9919	42.1174	0.000079

Coefficient of determination: The coefficient of determination or R^2 score denotes the variance. It is a measure of correlation. The greater the value, the better the results. Best possible score is 1.0 [28].

$$R^2(x, y) = 1 - \frac{\sum_{i=1}^{W}\sum_{j=1}^{H}(y_{ij} - \bar{x})^2}{\sum_{i=1}^{W}\sum_{j=1}^{H}(y_{ij} - \hat{x}_{ij})^2} \tag{5.8}$$

where, $\hat{x}$ is the predicted value, $\bar{x}$ is the mean value of all x.

5.2 Results

We evaluate the data deletion and reconstruction strategies on 3 different datasets including soil moisture, albedo, and Landsat satellite imagery.

5.2.1 Reconstruction for Grid Deletion

Table 5.1 shows the matrix reconstruction results by traditional machine learning algorithms. On the other hand, table 5.2 reports the evaluation metrics for the reconstructed visualizations by all the techniques we explained in Section 4.

Matrix: For matrix data reconstruction (Table 5.1), a different pattern for weather and satellite datasets is observed. For the weather dataset, multiquadrics interpolation seems to perform better in terms of all the evaluation metric. However, among regression models, SNN_k5_σ5 has lower error terms (both MSE and MAE) and a higher R^2 score. It is also observed that without additional features generated from the filtering, the initial four rows mostly fall into the light red zone in case of weather data. But, for satellite data, all the algorithms with features extracted from the 3×3 neighborhood falls into the green zone. The probable reason can be the presence of more diverse sizes and shapes in the satellite dataset than the weather dataset. With increased neighborhood, the features become more complex to model. Therefore, the performance is worsened for 5×5 neighborhood. Similarly, Shepard's IDW falls into the green zone, which implies that distance weighting has an impact on the satellite dataset.

> For weather data, SNN_k5_σ5 performs better than all other regression models in terms of all the evaluation metrics. But, for satellite imagery, all the regression models with features extracted from 3×3 neighborhood show high performance for all the evaluation metrics.

Image: Among the machine learning models, SNN_k3_σ3 performs better than the others for almost all datasets concerning all the evaluation metrics for image data as shown in Table 5.2. One exception is that SRGAN shows better performance in terms of PSNR and MSE for soil moisture data. Inpainting models, trained using both temporal and interpolated data shows better performance than all other methods. Even after grid deletion, there are enough available pixels in the surrounding for interpolation to fill in the deleted ones. there are slight differences in evaluation metrics among the inpainting models. The performance of these models is better because the inpainting GAN model is capable of generating high level features from input data and also recovering it. Moreover, the input being filled in, makes it easier for the model to reconstruct high quality data. As SRGAN upscales the low dimensional input space to high dimensional, the performance seems to be worse than inpainting networks.

Although these metrics are widely used for comparing image generation performances, they have some limitations in describing a method's visual quality [62, 107]. Therefore, the images need a further visual comparison.

> For both weather data and satellite imagery, Temp_Inpainting-Interp outperforms all other algorithms in terms of all the evaluation metrics and datasets.

Table 5.3: Performance comparison of deletion-reconstruction trade-offs (green indicates a better performance when comparing deletion techniques).

Dataset	Model			SRGAN			Image Inpainting		
	Deletion strategy	Deletion rate		SSIM	PSNR	MSE	SSIM	PSNR	MSE
		Mean	90th Percentile						
Soil moisture	Checkerboard	50%	50%	0.9241	27.4201	0.002	0.9977	45.143	0.00003
	Grid	75%	75%	0.8943	25.6858	0.0029	0.9949	41.7165	0.00007
	Variance based	76%	77%	0.8677	24.194	0.004	0.9948	41.5267	0.00006
Albedo	Checkerboard	50%	50%	0.8586	26.2088	0.0027	0.9909	39.1696	0.0001
	Grid	75%	75%	0.8242	24.324	0.00433	0.9801	35.1465	0.0004
	Variance based	77%	80%	0.8743	25.9852	0.003	0.9801	35.1465	0.0003
Satellite	Checkerboard	50%	50%	0.8262	21.0037	0.0157	0.9912	42.43858	0.00007
	Grid	75%	75%	0.7952	20.1861	0.0208	0.9909	41.9608	0.000008
	Variance based	76%	77%	0.8956	27.6901	0.0023	0.9298	32.7498	0.0005

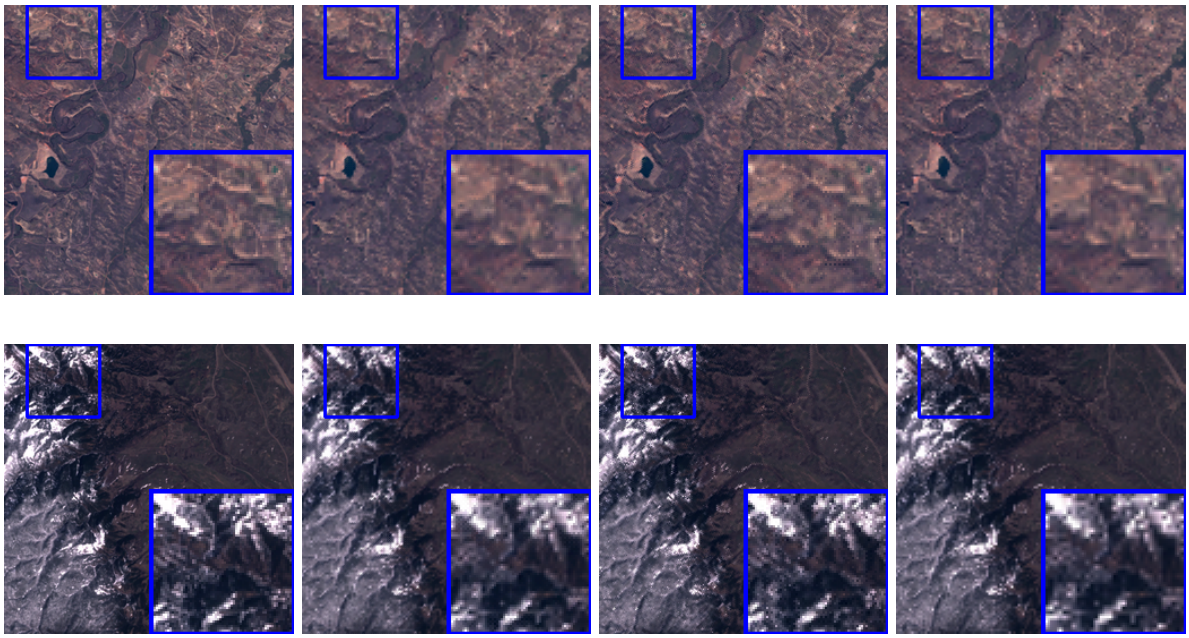
5.2.2 Visual Comparison

We compare the visual quality of the generated images for different methods. Fig. 5.5 and 5.6 illustrate some example outputs generated by various methods for the weather and satellite data.

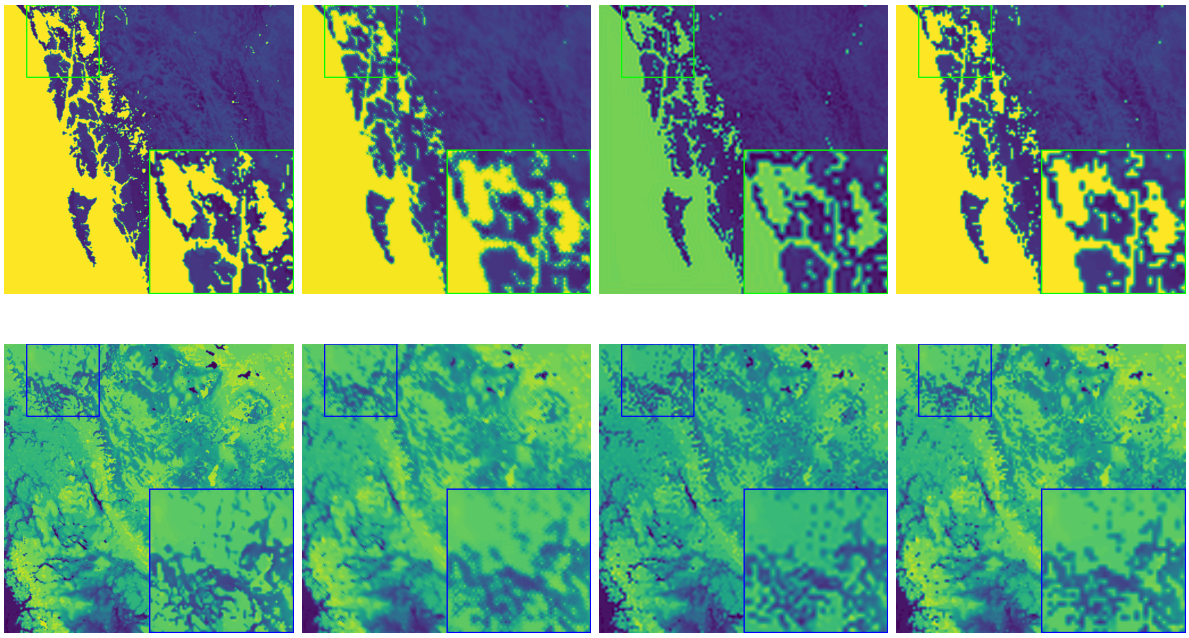
From Table 5.2, we can see that multiquadric and Shepard's interpolation performed quite well compared to other methods. But their visual quality is not pleasing (e.g., see Fig. 5.4). For instance, comparing with the original image, we can see color mismatch in the results of multiquadric interpolation for the weather dataset. Both regression and Shepard's interpolation shows blurry reconstruction for all the datasets. However, Shepard's interpolation has better recovery than SSN_k5_σ5 for the weather dataset.

We have used interpolation to fill in the unknown positions. Fig. 5.7 shows the failure of interpolation methods and the significant improvement obtained by our model Temp_Inpainting-Interp can do. As shown in this Fig. 5.7, the interpolated result generates a blurry image. This blurry image is upgraded by the inpainting model which is visually quite close to the original image.

The images generated by SNN and SRGAN are quite blurry compared to the original images. The images generated by the inpainting network show sharp edge boundaries (less blurriness) and better visual quality than that of SRGAN.

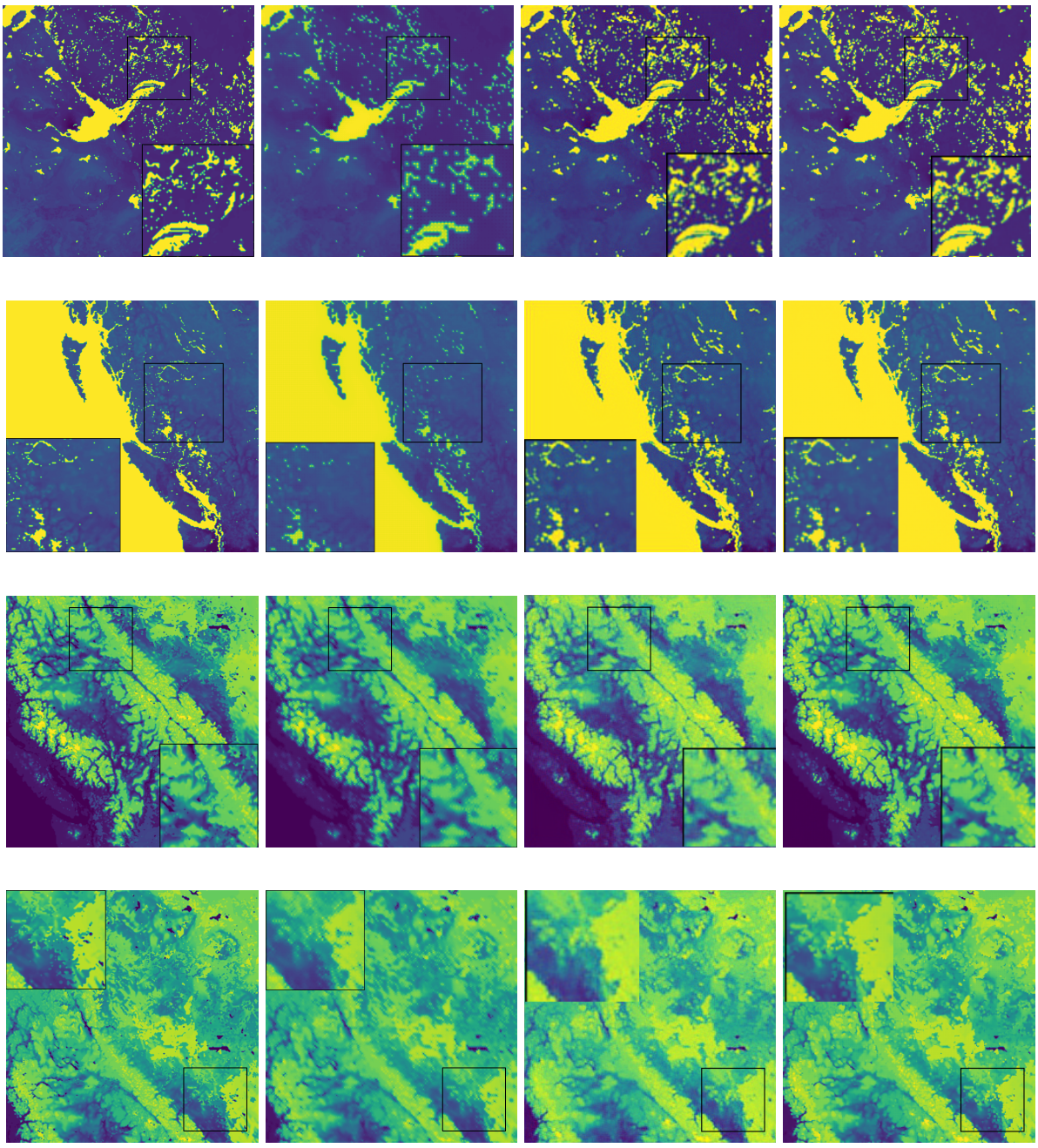


(a) Original image **(b)** SNN_k3_σ3 **(c)** Multiquadrics interpolation **(d)** Shepards' interpolation



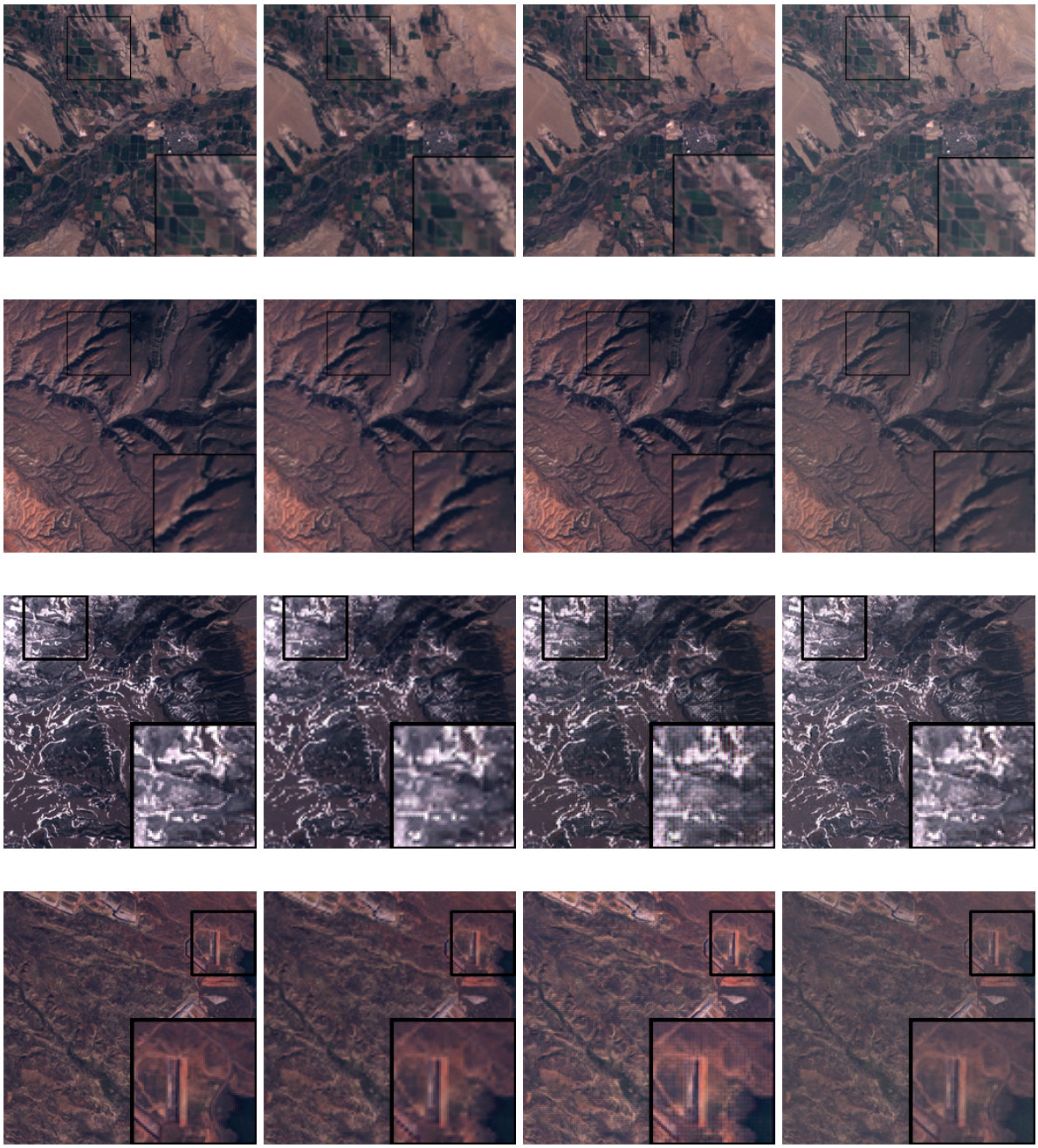
(e) Original Image **(f)** SNN_k5_σ5 **(g)** Multiquadric interpolation **(h)** Shepards' interpolation

Figure 5.4: Reconstruction results for matrix data after grid deletion. One can find the reconstruction problems by examining the annotated rectangles.



(a) Original Image **(b)** Baseline (SNN_k3_σ3) **(c)** SRGAN **(d)** Inpainting

Figure 5.5: Reconstruction results for weather dataset after grid deletion. The smaller square region is zoomed in and shown inside the larger square.



(a) Original Image **(b)** Baseline (SNN_k3_σ3) **(c)** SRGAN **(d)** Inpainting

Figure 5.6: Reconstruction results for satellite imagery after grid deletion. The smaller square region is zoomed in and shown inside the larger square.

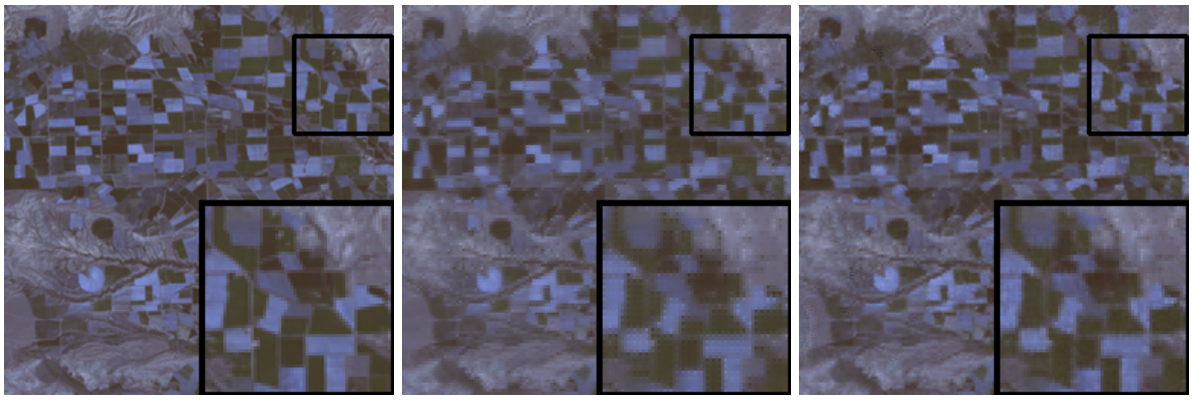


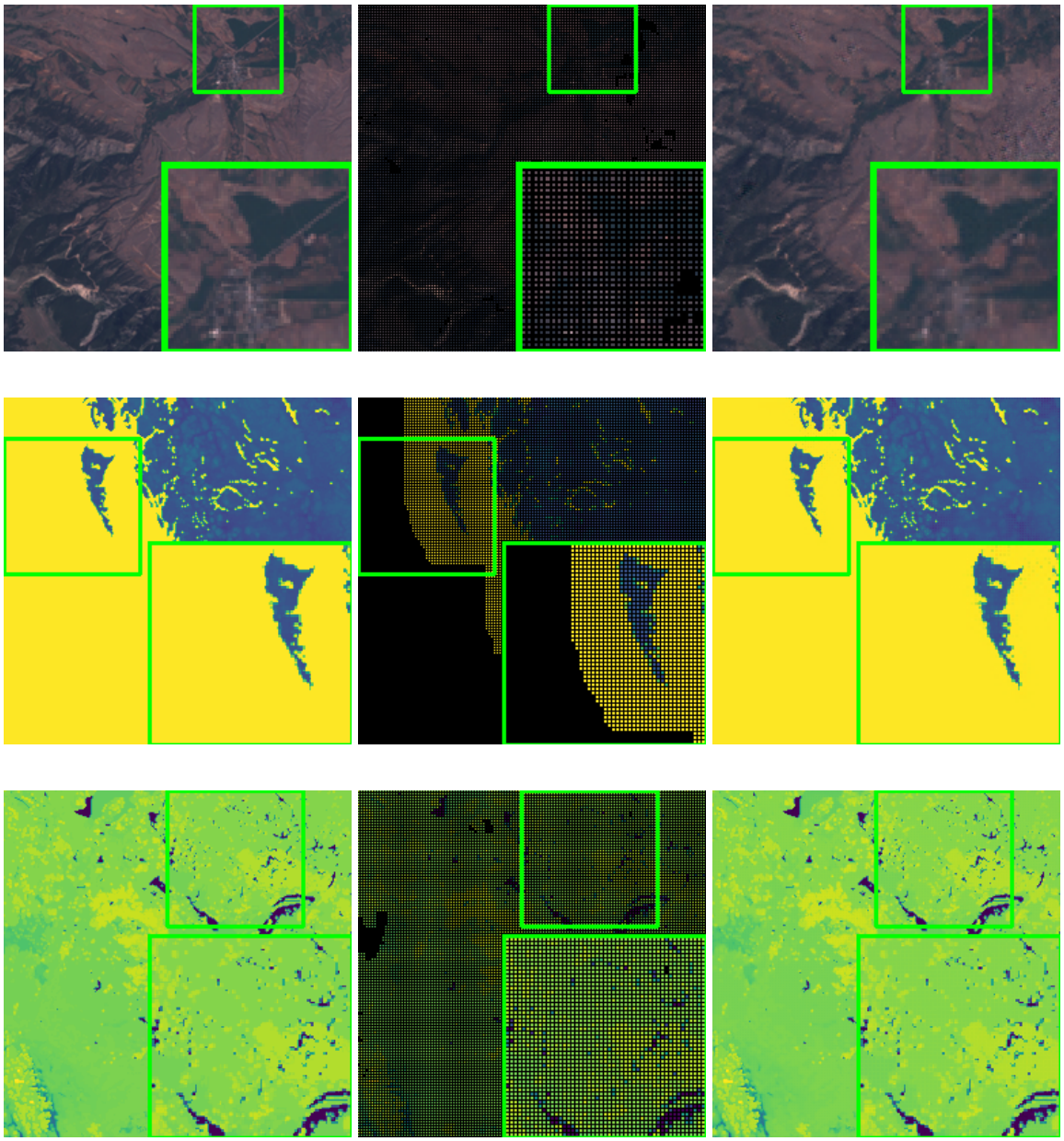
Figure 5.7: (left) Original image. (middle) Interpolated image (75%). (right) Reconstructed image.

5.2.3 Deletion-reconstruction trade-offs

Table 5.3 reports recovery performance for all the deletion methods. Following a checkerboard deletion, if the deletion rate is decreased (50%), the performance metrics do not improve significantly. We further investigate the additional deletion rate (higher than 75%) achieved through variance based deletion. The additional deletion rate varies based on the variance of the data. Both for soil moisture and satellite imagery, the variance based method increases deletion rate to 76% on an average. In some cases the deletion rate can reach over 77% (e.g., see Table 5.3). Since patterns and shapes that appear in satellite imagery are diverse, the overall deletion rate is lower than that of the weather dataset. The reconstruction accuracy is presumed to be lower in variance based deletion.

As shown in Table 5.2, the interpolation based approach outperformed the temporal data based recovery. This is because we only used small thresholds (i.e., R and Q equal to 20 and 10 percentiles, respectively) in our variance based deletion, which helps to keep the size of the deleted regions small. However, with larger thresholds when large regions containing prominent shapes get deleted, the interpolation cannot recover the shapes and the temporal data based model (or a model similar to [112]) performs better. However, the temporal model based approach requires retaining additional temporal data, yielding a low data reduction rate. Since we aimed for a high construction accuracy and large deletion rate, we suggest to use low thresholds for the variance base deletion, and thus the interpolation based technique appears to be a better choice for our problem domain. Fig. 5.8 shows the reconstruction result for variance based deletion using inpainting method. The recovered pixels on the grid are visually close to the original ones. The variance based deleted areas are also reconstructed well for the above mentioned reasons. As shown in Fig. 5.9, visual quality for reconstruction of grid deletion and checkerboard deletion is almost same.

The checkerboard deletion (50% deletion) improves the reconstruction accuracy only slightly when compared with the reconstruction performance over a grid deletion (75% deletion). For an additional deletion rate of 1% for soil moisture data, the reconstruction performance for image inpainting does



(a) Original image. **(b)** Variance based masked image. **(c)** Reconstructed image using inpainting.

Figure 5.8: Reconstruction results using inpainting model after variance based deletion.

Table 5.4: Performance comparison of lossy compression and grid deletion (green indicates a better performance).

Dataset	Model		Image Inpainting		
	mean	90th Percentile	SSIM	PSNR	MSE
Soil Moisture	75%	75%	0.9949	41.7165	0.00007
	75% compression by JPEG		0.8549	24.6065	0.0037
Albedo	75%	75%	0.9801	35.1465	0.0004
	75% compression by JPEG		0.8685	28.2629	0.002
satellite	75%	75%	0.9909	41.9608	0.000008
	75% compression by JPEG		0.8658	27.7016	0.0024

not seem to decrease noticeably. For satellite imagery, the decrease in SSIM and PSNR remains within 0.07 and 9.16, which indicates a high variance in the satellite imagery.

5.2.4 Lossy Compression

Since we are examining image based deletion and reconstruction, another viable option is to examine lossy image compression. We examined a lossy image compression [98], where the data files are stored in jpeg format with a size reduction of 70%. For reconstructing, we adopt our inpainting network. The observed SSIM, PSNR, MSE are as follows: 0.8549, 24.6065, 0.0037 in case of the compressed soil moisture data,.

This is a significant decrease in reconstruction performance in comparison with our inpainting method (while reconstructing from 75% deletion): 0.9948, 0.4152, and 0.00007. This is because lossy compression approximates all the data points. On the other hand, with our data deletion strategies, about 25% of the original data samples are always preserved. The visual quality is also poor due to this reason as shown in Fig. 5.9(top right). Hence, our deletion approaches seem to have more advantages over lossy compression when retaining some original data is preferred. A similar pattern is shown for other datasets in Table 5.4.

The lossy compression based inpainting performs poorly compared to the interpolation based inpainting.

5.2.5 Other reduction methods

In the original SRGAN work, the images are filtered with a Gaussian filter and downsampled by a factor of 4 [53]. They followed a traditional bicubic downsampling approach, which generates a downsampled image using cubic spline or polynomial technique [47]. Here, this approach modifies all the pixels of reduced image. Such traditional downsampling methods change the pixel value and provide only a modified sample instead of

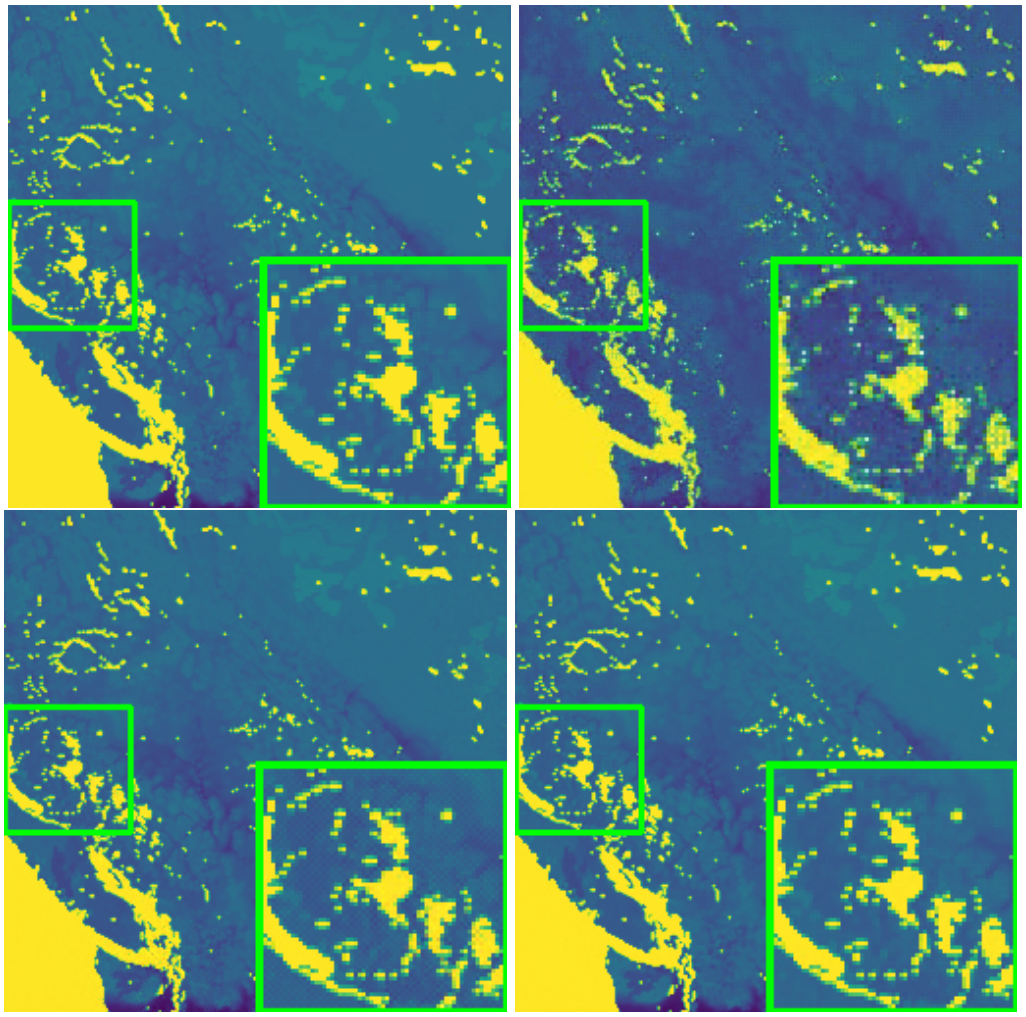


Figure 5.9: (top left) Original image. Reconstruction using inpainting from (top right) lossy compression, (bottom left) checkerboard deletion (75%), and (bottom right) grid deletion.

a true sample of the data. Similarly, the downsampled and lower level representation at the encoder output of the autoencoder architecture, are not suitable for data reduction. In the low dimensional representation, encoder does not keep any true data values [90].

5.3 Summary

In this chapter, we explained the experimental setup, datasets, evaluation metrics, and discussed the findings from the experiments. Among the deletion methods, grid deletion seems to be safe for visualization as it seems to have a considerable amount of reconstruction accuracy. It was seen that by decreasing the deletion rate to 50%, one can improve the accuracy only slightly. The variance based deletion has the advantage of having more than 75% deletion with visual and quantitative loss of information. Therefore, we suggest that, if the focus is to reconstruct matrix data with both good visual and quantitative quality, then SNN_k5_σ5

should be chosen for weather dataset and SNN_k3_σ3 for satellite imagery. On the other hand, for high quality image reconstruction, if temporal data is available, then Temp_Inpainting model should be used, as it guarantees to recover complex shape and proper color. The unavailability of temporal data can be handled with either Temp_Inpainting-Interp or Interp_Inpainting, as both shows comparable performance.

Chapter 6

Conclusion

6.1 Summary

In this work, we introduce a novel data reduction and recovery workflow that may improve the conventional compression approaches. Our deletion strategies show, for visualization purposes, one can aggressively reduce the geospatial plots and satellite imagery using uniform deletion (75% data points) and achieve a high reconstruction rate from the remaining 25% data points. We proposed a novel variance based technique that allows to further improve the deletion rate without compromising much with the reconstruction accuracy.

For all the deletion strategies, we examine a diverse set of machine learning techniques and adopt two deep learning models (SRGAN and image inpainting) for high-quality reconstruction. Our experimental results on different real-life datasets reveal that a reconstruction accuracy as high as 98.75% for geospatial data and 99.09% for satellite imagery can be achieved through an image inpainting method. We believe our approach will inspire further deep learning based research on data deletion and reconstruction for solving the problem of storage and transfer issues in big data management.

6.2 Contributions

Our main contributions are as follows:

1. A simple and easy-to-implement grid based data deletion strategy has been proposed. This deletion scheme provides an assurance of 75% data deletion. In addition to the 75% deletion rate, a novel reduction method based on variance is proposed, which guarantees additional 1% to 2% deletion rate. This further reduction does not compromise with the visual quality as well as maintaining accuracy comparable to that of 75% deletion with a decrease not more than 1% for spatial meteorological data and 7% for satellite imagery.

2. We have adopted GAN models for the recovery of the deleted data. There are two types of reconstruction models implemented. The first one is SRGAN that reconstructs the data from lower dimensional shape. The second one is the inpainting based recovery model. The maximum accuracy (SSIM) achieved was above 99% for the weather dataset and 98.1% for the satellite dataset by inpainting method which is a modification the work by Jihui et al [107].

Table 6.1: Performance comparison for places dataset

Model Name	SSIM	PSNR	MSE
SRGAN	0.9013	26.8061	0.0027
Image Inpainting	0.99	42.8321	0.00004

3. We also examined deletion-reconstruction trade off. Lowering the deletion rate from 75% does not have a significant impact on the evaluation metrics, which makes the grid deletion an optimal choice of deletion strategy. The variance based deletion increases the deletion rate more than 75% without losing important detail. The inpainting model is also able to reconstruct the deleted data by variance based deletion method with high accuracy 98.5% (average) for weather data and 92.98% for satellite imagery.

6.3 Limitation

In this section, we discuss some of the limitations of this work.

6.3.1 Data limitation

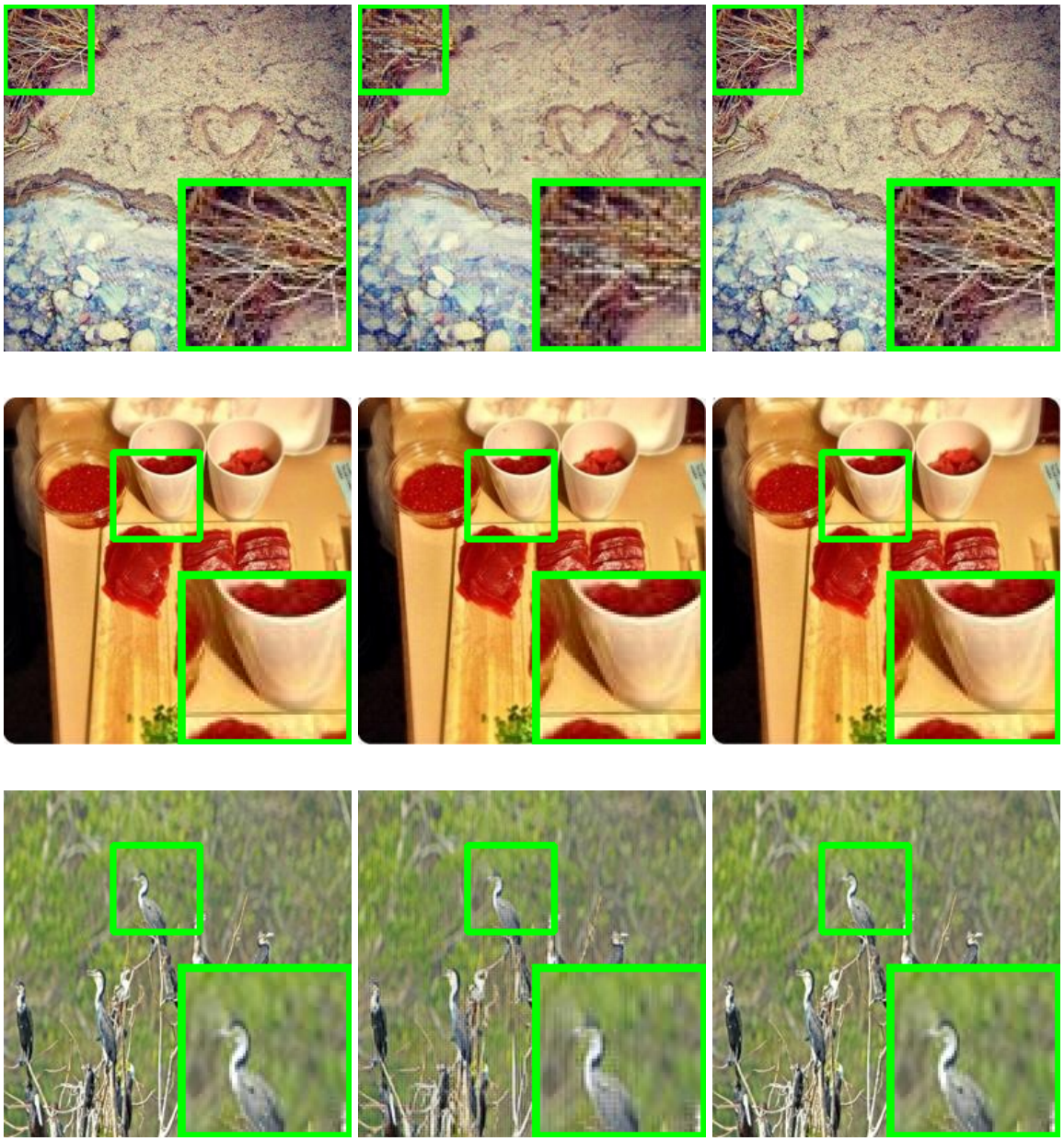
We run our experiments only on a meteorological dataset and satellite imagery. The analysis for other datasets could provide us more insights into generalizability.

Our work focuses on image data retrieval and leveraged the deep learning architectures used for image super resolution and image inpainting. However, they could not be applied directly to raw matrix data. All the inputs of these networks are of three channels and the way of processing an image is different from processing a matrix, especially for the weather dataset. Inpainting networks that work directly on the matrix can be an interesting avenue to explore.

We have chosen the Places365 dataset to make sure that our grid deletion pipeline works for the benchmark image dataset as well [113]. Table 6.1 shows the performance comparison for the places dataset. As we can see, the inpainting network trained with interpolated inputs seems to outperform SRGAN network. The visual comparison is shown in Fig. 6.1. We can also see that reconstruction by the inpainting model is better than SRGAN. This observation is consistent with all other findings.

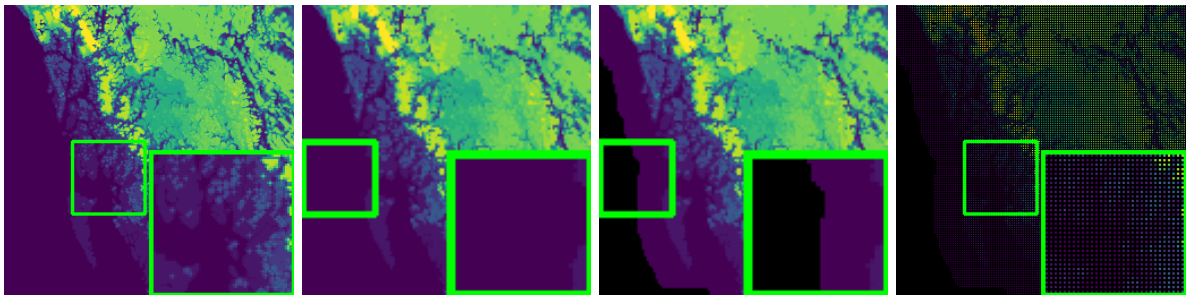
6.3.2 Color map problem

As mentioned in Section 5.1.1, we have used a perceptual color map for generating the contour plots. When we used a red, green, and blue color map, the result for SRGAN worsened. This indicates the impact of the color map on the reconstruction. Moreover, different color maps may suit different weather variables. This observation requires further in depth analysis.



(a) Original Image **(b)** SRGAN **(c)** Inpainting

Figure 6.1: Reconstruction results for places dataset after grid deletion.

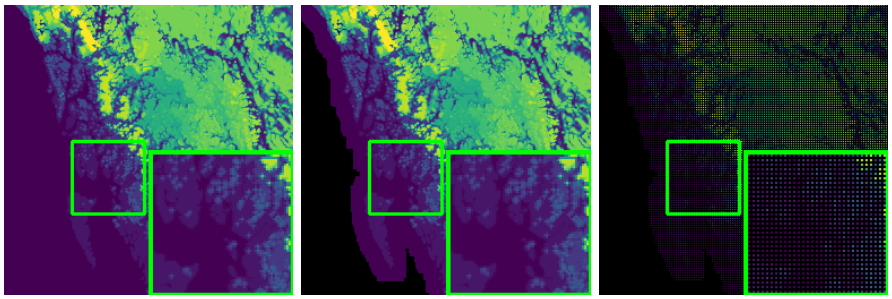


(a) Original Image

(b) Low dimensional image after grid deletion(75%)

(c) Variance based deletion (2.5%) on low dimensional image

(d) Overall deletion (77.51%)



(e) Original Image

(f) Variance based deletion (15.50%)

(g) Overall deletion after adding grid deletion (78.88%)

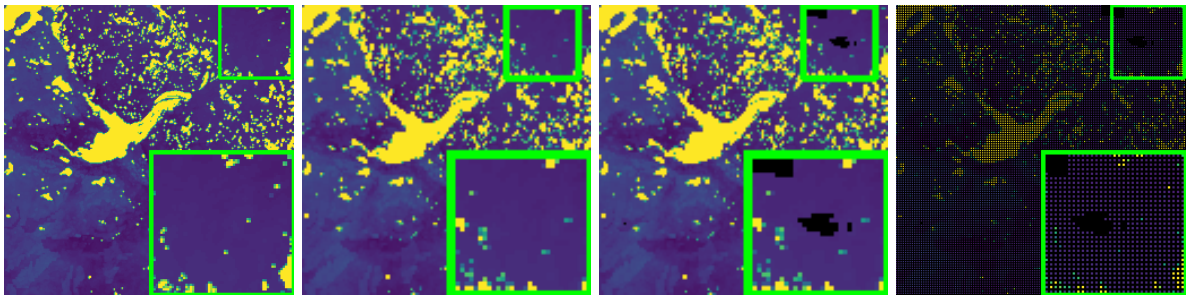
Figure 6.2: (first row) Illustration of steps for grid deletion and then variance based deletion. (second row) Illustration of steps for variance based deletion and then grid deletion for low variance data.

6.3.3 Grid+variance vs variance+grid deletion

In our variance based deletion, we first deleted the data samples in a grid then applied the variance based deletion on it. We also alter the sequence by applying variance based deletion first and then grid deletion. Fig. 6.2 and 6.3 shows the comparison of altering the sequence. As we can see, applying variance based deletion first provides 15.50% deletion which is bigger than applying the variance based deletion later. It would be interesting to investigate whether this observation can be leveraged to obtain a better deletion-reconstruction trade off.

6.3.4 Evaluation metrics

We adopted widely used evaluation metrics, e.g. PSNR, SSIM, MSE, MAE. These metrics are capable of showing a quantitative comparison. However, image reconstruction requires a better perceptual comparison by a human. A probable metric can be Mean opinion score (MOS) [53, 87]. *Opinion score* is defined by the value, assigned to a performance of a system, according to a predefined scale by a subject from his point of view. The mean opinion score is the average of all these values.

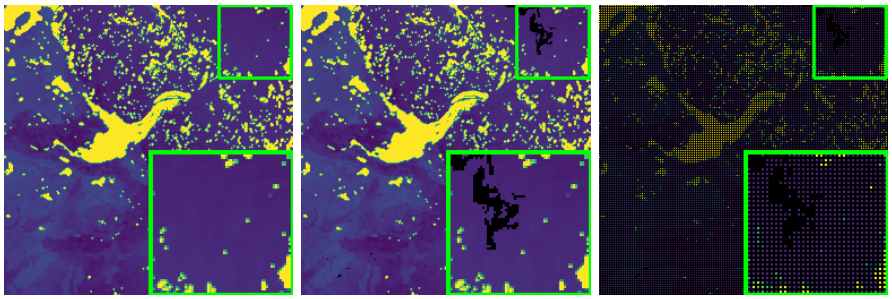


(a) Original Image

(b) Low dimensional image after grid deletion(75%)

(c) Variance based deletion (0.13%) on low dimensional image

(d) Overall deletion (75.13%)



(e) Original Image

(f) Variance based deletion (64.39%)

(g) Overall deletion adding grid deletion (75.16%)

Figure 6.3: (first row) Illustration of steps for grid deletion and then variance based deletion. (second row) Illustration of steps for variance based deletion and then grid deletion for high variance data.

6.4 Future Work

In this research, we proposed data deletion approaches, which allow the user to store the reduced version of the data with low storage space and transfer it to a local machine with low bandwidth. We also presented trained models that can recover image data with high accuracy that can be used for visual analysis. As mentioned in Section 6.3, there are a few limitations that point to the following future works. The following are some directions for future research.

1. The deep learning models are performing well on the image form of data. Can a recovery model be built that can leverage both matrix and image data (Fig.6.4)?

2. The current loss for SRGAN is based on JS-divergence, which makes it difficult to optimize. Can we improve SRGAN performance by implementing a WGAN-GP loss?

3. A temporal based data deletion strategy can be implemented. For this, a sequential model can be developed that understands the pattern between temporal interval and thus will be able to select indices to be deleted. This may eliminate the necessity of temporal dependency.

4. The mask generated by variance based deletion is random. Can partial convolutions be implemented in

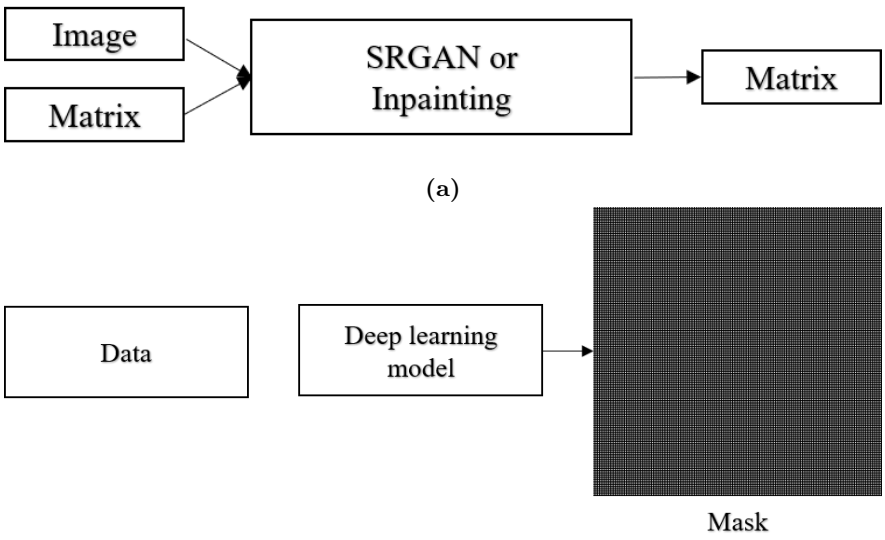


Figure 6.4: Illustration of (a) A recovery model that can leverage both matrix and image data, (b) Auto selected mask.

the network to deal with the randomness and eliminate the necessity of interpolation as a preprocessing step and also achieve high accuracy?

5. Can a mask, indicating the samples to be deleted, be designed using deep learning architecture like an auto selection of mask (Fig. 6.4)?

6. An autoencoder based model can be used for both deletion and reconstruction. In traditional autoencoders, the encoded data is a representation and completely differs from the original data samples. Along with the error term between the input of encoder and output of the decoder, a new loss function can be incorporated based on the error term between the lower dimensional representation and lower dimensional data obtained by the deletion method. Consequently, the lower dimensional representation may have less loss of information.

7. The deletion is done on red, green, and blue channels of satellite data. It may be possible to examine other spectral channels. Currently, the reconstruction model is not able to work on any other channel than the above mentioned ones. So designing a reconstruction model that can work regardless of the specified channel would be interesting.

References

[1] Landsat 7 ETM+ image courtesy of the U.S. geological survey. data retrieved from USGS earthexplorer, https://earthexplorer.usgs.gov/.

[2] A. Aidini, G. Tsagkatakis, and P. Tsakalides. Tensor dictionary learning with representation quantization for remote sensing observation compression. In *2020 Data Compression Conference (DCC)*, pages 283–292. IEEE, 2020.

[3] S. Albawi, T. A. Mohammed, and S. Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.

[4] S. Andrefouet, R. Bindschadler, E. Brown de Colstoun, M. Choate, W. Chomentowski, J. Christopherson, B. Doorn, D. Hall, C. Holifield, S. Howard, et al. Preliminary assessment of the value of landsat-7 etm+ data following scan line corrector malfunction. *US Geological Survey, EROS Data Center: Sioux Falls, SD, USA*, 2003.

[5] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[6] S. Arora and G. Kumar. Review of image compression techniques. *International Journal of Recent Research Aspects*, 5:185–188, 2018.

[7] J. Ballé, V. Laparra, and E. P. Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016.

[8] M. Banday, P. G. Scholar, R. Sharma, and D. Dehradun. Image inpainting – an inclusive review of the underlying algorithm and comparative study of the associated techniques.

[9] R. Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[10] M. Bertalmio, A. L. Bertozzi, and G. Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I. IEEE, 2001.

[11] C. M. Bishop. *Pattern Recognition and Machine Learning by Christopher M. Bisho*. Springer Science+ Business Media, LLC, 2006.

[12] K. Black. *Business statistics: for contemporary decision making*. John Wiley & Sons, 2019.

[13] G.-P. Bonneau, H.-C. Hege, C. R. Johnson, M. M. Oliveira, K. Potter, P. Rheingans, and T. Schultz. Overview and state-of-the-art of uncertainty visualization. In *Scientific Visualization*, pages 3–27. Springer, 2014.

[14] A. C. Bovik. *Handbook of image and video processing*. Academic press, 2010.

[15] S. Brown, R. Tauler, and B. Walczak. *Comprehensive chemometrics: chemical and biochemical data analysis*. Elsevier, 2020.

[16] A. Cedilnik, B. Geveci, K. Moreland, J. P. Ahrens, and J. M. Favre. Remote large data visualization in the paraview framework. In *EGPGV*, pages 163–170, 2006.

[17] S. Chandak, K. Tatwawadi, C. Wen, L. Wang, J. Aparicio Ojea, and T. Weissman. LFZip: Lossy compression of multivariate floating-point time series data via improved prediction. In *2020 Data Compression Conference (DCC)*, pages 342–351, 2020.

[18] Y. Chen, L. Tang, X. Yang, R. Fan, M. Bilal, and Q. Li. Thick clouds removal from multitemporal zy-3 satellite images using deep learning. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 13:143–153, 2019.

[19] S. Cheng and F. Lu. A two-step method for missing spatio-temporal data reconstruction. *ISPRS International Journal of Geo-Information*, 6(7):187, 2017.

[20] M. E. Chenoweth and S. A. Sarra. A numerical study of generalized multiquadric radial basis function interpolation. *SIAM Undergraduate Research Online*, 2(2):58–70, 2009.

[21] A. Cotter, H. Jiang, and K. Sridharan. Two-player games for efficient non-convex constrained optimization. In *Algorithmic Learning Theory*, pages 300–332. PMLR, 2019.

[22] S. Das and P. N. Rao. Principal component analysis based compression scheme for power system steady state operational data. In *ISGT2011-India*, pages 95–100. IEEE, 2011.

[23] E. Diao, J. Ding, and V. Tarokh. DRASIC: distributed recurrent autoencoder for scalable image compression. In *2020 Data Compression Conference (DCC)*, pages 3–12, 2020.

[24] E. Diao, J. Ding, and V. Tarokh. Drasic: Distributed recurrent autoencoder for scalable image compression. In *2020 Data Compression Conference (DCC)*, pages 3–12. IEEE, 2020.

[25] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.

[26] J. Dong, R. Yin, X. Sun, Q. Li, Y. Yang, and X. Qin. Inpainting of remote sensing sst images with deep convolutional generative adversarial network. *IEEE Geoscience and Remote Sensing Letters*, 16(2):173–177, 2018.

[27] E. Doutsi and P. Tsakalides. Image compression based on neuroscience models: Rate-distortion performance of the neural code. In *2020 Data Compression Conference (DCC)*, pages 364–364. IEEE, 2020.

[28] N. R. Draper and H. Smith. *Applied regression analysis*, volume 326. John Wiley & Sons, 1998.

[29] C. K. Enders. *Applied missing data analysis*. Guilford press, 2010.

[30] A. Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. " O'Reilly Media, Inc.", 2017.

[31] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[32] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

[33] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016.

[34] A. N. Habermann. Automatic deletion of obsolete information. *Journal of Systems and Software*, 5(2):145–154, 1985.

[35] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[36] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets*, pages 286–297. Springer, 1990.

[37] A. Horé and D. Ziou. Image quality metrics: PSNR vs. SSIM. In *2010 20th International Conference on Pattern Recognition*, pages 2366–2369, 2010.

[38] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG)*, 36(4):1–14, 2017.

[39] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[40] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[41] F. A. Jassim. Image inpainting by kriging interpolation technique. *arXiv preprint arXiv:1306.0139*, 2013.

[42] J. R. Jensen et al. *Introductory digital image processing: a remote sensing perspective.* Number Ed. 2. Prentice-Hall Inc., 1996.

[43] F. Jiang, W. Tao, S. Liu, J. Ren, X. Guo, and D. Zhao. An end-to-end compression framework based on convolutional neural networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(10):3007–3018, 2017.

[44] N. Johnston, D. Vincent, D. Minnen, M. Covell, S. Singh, T. Chinen, S. Jin Hwang, J. Shor, and G. Toderici. Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4385–4393, 2018.

[45] M. Jones and J. A. Rice. Displaying the important features of large collections of similar curves. *The American Statistician*, 46(2):140–145, 1992.

[46] A. Kadurin, S. Nikolenko, K. Khrabrov, A. Aliper, and A. Zhavoronkov. drugan: an advanced generative adversarial autoencoder model for de novo generation of new molecules with desired molecular properties in silico. *Molecular pharmaceutics*, 14(9):3098–3104, 2017.

[47] R. Keys. Cubic convolution interpolation for digital image processing. *IEEE transactions on acoustics, speech, and signal processing*, 29(6):1153–1160, 1981.

[48] D. Kim, S. Woo, J. Lee, and I. S. Kweon. Recurrent temporal aggregation framework for deep video inpainting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(5):1038–1052, 2020.

[49] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[50] K. Knight. *Mathematical Statistics*. Chapman and Hall, New York, 2000.

[51] S. Kotsiantis and D. Kanellopoulos. Discretization techniques: A recent survey. *GESTS International Transactions on Computer Science and Engineering*, 32(1):47–58, 2006.

[52] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[53] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[54] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang. Learning convolutional networks for content-weighted image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3214–3223, 2018.

[55] Y. Li, Z. Li, Z. Zhang, L. Chen, S. Kurkute, L. Scaff, and X. Pan. High-resolution regional climate modeling and projection over western canada using a weather research forecasting model with a pseudo-global warming approach. *Hydrology and Earth System Sciences*, 23(11):4635–4659, 2019.

[56] L. Liao, J. Xiao, Y. Li, M. Wang, and R. Hu. Learned representation of satellite image series for data compression. *Remote Sensing*, 12(3):497, 2020.

[57] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 85–100, 2018.

[58] H. Liu, F. Hussain, C. L. Tan, and M. Dash. Discretization: An enabling technique. *Data mining and knowledge discovery*, 6(4):393–423, 2002.

[59] D. Lowd and P. Domingos. Naive bayes models for probability estimation. In *Proceedings of the 22nd international conference on Machine learning*, pages 529–536, 2005.

[60] J. Lowe and M. Matthee. Requirements of data visualisation tools to analyse big data: A structured literature review. In *Conference on e-Business, e-Services and e-Society*, pages 469–480. Springer, 2020.

[61] D. J. MacKay. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992.

[62] H. Maître. *From photon to pixel: the digital camera handbook*. John Wiley & Sons, 2017.

[63] Y. Marchetti, H. Nguyen, A. Braverman, and N. Cressie. Spatial data compression via adaptive dispersion clustering. *Computational Statistics & Data Analysis*, 117:138–153, 2018.

[64] R. Mehra, N. Bhatt, F. Kazi, and N. Singh. Analysis of PCA based compression and denoising of smart grid data under normal and fault conditions. In *2013 IEEE International Conference on Electronics, Computing and Communication Technologies*, pages 1–6. IEEE, 2013.

[65] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool. Conditional probability models for deep image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4394–4402, 2018.

[66] K. Nazeri, E. Ng, T. Joseph, F. Z. Qureshi, and M. Ebrahimi. Edgeconnect: Generative image inpainting with adversarial edge learning. *arXiv preprint arXiv:1901.00212*, 2019.

[67] R. M. Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[68] J. R. Nuñez, C. R. Anderton, and R. S. Renslow. Optimizing colormaps with consideration for color vision deficiency to enable accurate interpretation of scientific data. *PloS one*, 13(7):e0199239, 2018.

[69] S. Oehmcke and F. Gieseke. Learning selection masks for deep neural networks. *arXiv preprint arXiv:1906.04673*, 2019.

[70] Y. Pan, F. Zhu, T. Gao, and H. Yu. Adaptive deep learning based time-varying volume compression. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1187–1194. IEEE, 2019.

[71] V. C. Pandey and K. Bauddh. *Phytomanagement of Polluted Sites: Market Opportunities in Sustainable Phytoremediation*. Elsevier, 2018.

[72] A. Pang, J. J. Furman, and W. Nuss. Data quality issues in visualization. In *Visual Data Exploration and Analysis*, volume 2178, pages 12–23. International Society for Optics and Photonics, 1994.

[73] Q. A. Panhwar, A. Ali, U. A. Naher, and M. Y. Memon. Fertilizer management strategies for enhancing nutrient use efficiency and sustainable wheat production. In *Organic Farming*, pages 17–39. Elsevier, 2019.

[74] P. Parekar and S. Thakare. Lossless data compression algorithm–a review. *International Journal of Computer Science & Information Technologies*, 5(1), 2014.

[75] 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*, pages 2536–2544, 2016.

[76] A. Rap, S. Ghosh, and M. H. Smith. Shepard and Hardy Multiquadric Interpolation Methods for Multicomponent Aerosol–Cloud Parameterization. *Journal of the Atmospheric Sciences*, 66(1):105–115, 01 2009.

[77] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[78] H. Sak, A. W. Senior, and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. 2014.

[79] K. Satone, A. Deshmukh, and P. Ulhe. A review of image compression techniques. In *2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)*, volume 1, pages 97–101. IEEE, 2017.

[80] P. Scaramuzza and J. Barsi. Landsat 7 scan line corrector-off gap-filled product development. In *Proceeding of Pecora*, volume 16, pages 23–27, 2005.

[81] H. Shen, X. Li, Q. Cheng, C. Zeng, G. Yang, H. Li, and L. Zhang. Missing information reconstruction of remote sensing data: A technical review. *IEEE Geoscience and Remote Sensing Magazine*, 3(3):61–85, 2015.

[82] M. J. Shensa. The discrete wavelet transform: wedding the a trous and mallat algorithms. *IEEE Transactions on signal processing*, 40(10):2464–2482, 1992.

[83] D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 ACM National Conference*, pages 517–524, 1968.

[84] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[85] M. Singh, S. Kumar, S. Singh, and M. Shrivastava. Various image compression techniques: Lossy and lossless. *International Journal of Computer Applications*, 142(6):23–26, 2016.

[86] W. C. Skamarock. A description of the advanced research wrf version 2, ncar technical note, ncar/tn-468+ str. *http://www. mmm. ucar. edu/wrf/users/docs/arw_v2. pdf*, 2005.

[87] R. C. Streijl, S. Winkler, and D. S. Hands. Mean opinion score (mos) revisited: methods and applications, limitations and alternatives. *Multimedia Systems*, 22(2):213–227, 2016.

[88] P. Strzelczak, E. Adamczyk, U. Herman-Izycka, J. Sakowicz, L. Slusarczyk, J. Wrona, and C. Dubnicki. Concurrent deletion in a distributed content-addressable storage system with global deduplication. In *Presented as part of the 11th {USENIX} Conference on File and Storage Technologies ({FAST} 13)*, pages 161–174, 2013.

[89] J. Tasnim and D. Mondal. Data reduction and deep-learning based recovery for geospatial visualization and satellite imagery. In *2020 IEEE International Conference on Big Data (Big Data)*, 2020. Accepted, In press.

[90] L. Theis, W. Shi, A. Cunningham, and F. Huszár. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*, 2017.

[91] L. Theis, W. Shi, A. Cunningham, and F. Huszár. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*, 2017.

[92] M. E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of machine learning research*, 1(Jun):211–244, 2001.

[93] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar. Variable rate image compression with recurrent neural networks. *arXiv preprint arXiv:1511.06085*, 2015.

[94] G. Toderici, D. Vincent, N. Johnston, S. Jin Hwang, D. Minnen, J. Shor, and M. Covell. Full resolution image compression with recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5306–5314, 2017.

[95] S. Van Buuren. *Flexible Imputation of Missing Data*. Chapman & Hall/CRC Interdisciplinary Statistics Series, 2012.

[96] L. Van Der Maaten, E. Postma, and J. Van den Herik. Dimensionality reduction: a comparative. *J Mach Learn Res*, 10(66-71):13, 2009.

[97] R. Viviani, G. Grön, and M. Spitzer. Functional principal component analysis of fmri data. *Human brain mapping*, 24(2):109–129, 2005.

[98] G. K. Wallace. The JPEG still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

[99] H.-J. Wang, J.-L. Wang, M.-H. Wang, and Y.-M. Yin. Efficient image inpainting based on bilinear interpolation downscaling. *Guangxue Jingmi Gongcheng(Optics and Precision Engineering)*, 18(5):1234–1241, 2010.

[100] Z. Wang, J. Chen, and S. C. Hoi. Deep learning for image super-resolution: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[101] Wikipedia contributors. File deletion — Wikipedia, the free encyclopedia, 2020. [Online; accessed 15-September-2020].

[102] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *Advances in neural information processing systems*, pages 341–349, 2012.

[103] X. Xu, T. Liang, J. Zhu, D. Zheng, and T. Sun. Review of classical dimensionality reduction and sample selection methods for large-scale data processing. *Neurocomputing*, 328:5–15, 2019.

[104] Z. Yan, X. Li, M. Li, W. Zuo, and S. Shan. Shift-net: Image inpainting via deep feature rearrangement. In *Proceedings of the European conference on computer vision (ECCV)*, pages 1–17, 2018.

[105] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. *arXiv preprint arXiv:1703.10847*, 2017.

[106] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[107] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Generative image inpainting with contextual attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5505–5514, 2018.

[108] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Free-form image inpainting with gated convolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4471–4480, 2019.

[109] Y. Yu. Functional principal component analysis: A robust method for time-series phenotypic data, 2020.

[110] C. Zeng, H. Shen, and L. Zhang. Recovering missing pixels for landsat etm+ slc-off imagery using multi-temporal regression analysis and a regularization method. *Remote Sensing of Environment*, 131:182–194, 2013.

[111] Y. Zeng, J. Fu, and H. Chao. Learning joint spatial-temporal transformations for video inpainting. *arXiv preprint arXiv:2007.10247*, 2020.

[112] Q. Zhang, Q. Yuan, C. Zeng, X. Li, and Y. Wei. Missing data reconstruction in remote sensing image with a unified spatial–temporal–spectral deep convolutional neural network. *IEEE Transactions on Geoscience and Remote Sensing*, 56(8):4274–4288, 2018.

[113] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[114] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

Appendix A

Details of the model architectures explained in Section 4

Table A.1: Details about generator architecture for SRGAN for Fig. 4.8.

	Layer Name	Layer Type	Arguments
Pre-residual block	Input Layer	Input	Shape: 128, 128, 3
	Conv_1	Conv	A: Parametric ReLU, P: same, K: 9, O: 64
16 Residual Blocks	Conv_Res_1	Conv	A: Parametric ReLU, P: same, K: 9, O: 64
	Batch_Norm_1	BatchNorm	Momentum=0.8
	Conv_Res_2	Conv	A: None, P: same, K: 3, O: 64
	Batch_Norm_2	BatchNorm	Momentum=0.8
	Res_Connection_1	Add()	Input to Conv_Res_1 and output of Batch_Norm_2
Post-residual block	Conv_2	Conv	A: Parametric ReLU, P: same, K: 3, O: 64
	Res_Connection_2	Add()	Input to Residual Block and output of Conv_2
	UpSampling2D_1	Up Sampling	Factor: 2
	Conv_3	Conv	A: Parametric ReLU, P: same, K: 3, O: 256
Output layer	Conv_4	Conv	A: tanh, P: same, K: 9 O: 3

Table A.2: Details about recovery network architecture for inpainting network without contextual layer for Fig. 4.10.

Layer Name	Layer Type	Arguments
Input	Convolution	Shape: 256, 256, 3
Conv_1	Convolution	K: 5, S: 1, O: 32
Conv_2	Convolution	K: 3, S: 2, O: 64
Conv_3	Convolution	K: 3, S: 1, O: 64
Conv_4	Convolution	K: 3, S: 2, O: 128
Conv_5	Convolution	K: 3, S: 1, O: 128
Conv_6	Convolution	K: 3, S: 1, O: 128
Dilated_conv_1	Dilated Convolution	K: 3, D: 2, S: 1, O: 128
Dilated_conv_1	Dilated Convolution	K: 3, D: 4, S: 1, O: 128
Dilated_conv_1	Dilated Convolution	K: 3, D: 8, S: 1, O: 128
Dilated_conv_1	Dilated Convolution	K: 3, D: 16, S: 1, O: 128
Conv_7	Convolution	K: 3, S: 1, O: 128
Conv_8	Convolution	K: 3, S: 1, O: 128
Conv_9	Convolution	K: 3, S: 1, O: 64
Conv_10	Convolution	K: 3, S: 1, O: 64
Conv_11	Convolution	K: 3, S: 1, O: 32
Conv_12	Convolution	K: 3, S: 1, O: 16
Output	Convolution	K: 3, S: 1, O: 3

Table A.3: Details about discriminator architecture for SRGAN for Fig. 4.8.

Layer Name	Layer Type	Arguments
Input	Input	Shape: 256, 256, 3
Conv_1	Conv	P: same, k: 3, O: 64, S: 1
Activation_layer_1	Activation	Leaky ReLU, alpha=0.2
Conv_2	Conv	P: same, k: 3, O: 64, S: 2
Activation_layer_2	Activation	Leaky ReLU, alpha=0.2
Batch_Norm_1	BatchNorm	Momentum=0.8
Conv_3	Conv	P: same, k: 3, O: 128, S: 1
Activation_layer_3	Activation	Leaky ReLU, alpha=0.2
Batch_Norm_2	BatchNorm	Momentum=0.8
Conv_4	Conv	P: same, k: 3, O: 128, S: 2
Activation_layer_4	Activation	Leaky ReLU, alpha=0.2
Batch_Norm_3	BatchNorm	Momentum=0.8
Conv_5	Conv	P: same, k: 3, O: 256, S: 1
Activation_layer_5	Activation	Leaky ReLU, alpha=0.2
Batch_Norm_4	BatchNorm	Momentum=0.8
Conv_6	Conv	P: same, k: 3, O: 256, S: 2
Activation_layer_6	Activation	Leaky ReLU, alpha=0.2
Batch_Norm_5	BatchNorm	Momentum=0.8
Conv_7	Conv	P: same, k: 3, O: 512, S: 1
Activation_layer_7	Activation	Leaky ReLU, alpha=0.2
Batch_Norm_6	BatchNorm	Momentum=0.8
Conv_8	Conv	P: same, k: 3, O: 512, S: 2
Activation_layer_8	Activation	Leaky ReLU, alpha=0.2
Batch_Norm_7	BatchNorm	Momentum=0.8
Dense_2	Dense	Neuron: 1024
Activation_layer_9	Activation	Leaky ReLU, alpha=0.2
Dense_3	Dense	Neuron: 1, Activation: Sigmoid

Table A.4: Layer description of contextual attention pipeline for Fig. 4.10.

Layer Name	Layer Type	Arguments
Input	Convolution	Shape: 256, 256, 3
Context_conv_1	Convolution	K: 5, S: 1, O: 32
Context_conv_2	Convolution	K: 3, S: 2, O: 64
Context_conv_3	Convolution	K: 3, S: 1, O: 64
Context_conv_4	Convolution	K: 3, S: 2, O: 128
Context_conv_5	Convolution	K: 3, S: 1, O: 128
Context_conv_6	Convolution	K: 3, S: 1, O: 128
Contextual layer	Conv, Conv, Softmax, Deconv	K: 3, D: 2, S: 1, O: 128
Context_conv_7	Convolution	K: 3, S: 1, O: 128
Context_conv_8	Convolution	K: 3, S: 1, O: 128
Concat	Concat	Context_conv_8 and Conv_6 of main autoencoder network

Table A.5: Details about local critic architecture for inpainting network for Fig. 4.10.

Layer Name	Layer Type	Arguments
Input	Convolution	Shape: 256, 256, 3
Conv_1	Convolution	K: 5, S: 2, O: 64
Conv_2	Convolution	K: 5, S: 2, O: 128
Conv_3	Convolution	K: 5, S: 2, O: 256
Conv_4	Convolution	K: 3, S: 2, O: 512
Output	Fully connected	1

Table A.6: Details about global critic architecture for inpainting network for Fig. 4.10.

Layer Name	Layer Type	Arguments
Input	Convolution	Shape: 256, 256, 3
Conv_1	Convolution	K: 5, S: 2, O: 64
Conv_2	Convolution	K: 5, S: 2, O: 128
Conv_3	Convolution	K: 5, S: 2, O: 256
Conv_4	Convolution	K: 3, S: 2, O: 256
Output	Fully connected	1

Appendix B

Implementation details of Table. 3.1

We generated one dimensional normal distribution with the python code $numpy.default\_rng().normal(loc, scale, size)$. Here, loc is the mean of the distribution, $scale$ is the standard deviation and $size$ is the output shape of the data. The code generates the data samples with given mean and standard deviations. We had generated one dimensional normal distribution with two sets of mean and standard deviations: one is mean, standard deviation $= 4.45, 7$ and the other one is mean, standard deviation $= 4.45, 2.93$. $Size$ of the generated data was kept one dimensional with the value 100. We scaled the values between zero and one. The mean and standard deviation were kept close to that of the datasets that we used in our experiments. We applied our variance based deletion approach with a range of possible values of k, R, and Q. For one dimensional data, neighbourhood area k is defined by the k number of data samples on both sides of a particular data sample.