Blind Inpainting and Super-resolution Using Convolutional Neural Netwok

by

SURABHI SOHONEY
ID: 201411027

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

MASTER OF TECHNOLOGY
in
INFORMATION AND COMMUNICATION TECHNOLOGY
to

DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY

May, 2016

Declaration

I hereby declare that

i) the thesis comprises of my original work towards the degree of Master of Technology in Information and Communication Technology at Dhirubhai Ambani Institute of Information and Communication Technology and has not been submitted elsewhere for a degree,

ii) due acknowledgment has been made in the text to all the reference material used.

SURABHI SOHONEY

Certificate

This is to certify that the thesis work entitled **Blind Inpainting and Super-resolution Using Convolutional Neural Network** has been carried out by Surabhi Sohoney for the degree of Master of Technology in Information and Communication Technology at *Dhirubhai Ambani Institute of Information and Communication Technology* under my/our supervision.

Prof. Manjunath V. Joshi
Thesis Supervisor

Acknowledgments

First and foremost, I take this opportunity to express a deep sense of gratitude towards my guide Prof. Manjunath V. Joshi for his support with his patience, knowledge and inspiration throughout the research work. Without his constructive criticism, this work would never have been the same as it is now. With his guidance, my research and thesis writing work were carried out very efficiently.

Besides my supervisor, I would also like to thank the panel members Prof. Hemant Patil, Prof. Asim Banerjee, Prof. Nagraj Ramarao and Prof. Sanjay Srivastava for their insightful comments and valuable inputs.

I would also like to thank my colleagues, Prathmesh, Ketul, Hanish and Meet for their valuable suggestions and helpful discussions. I would like to thank Mr. Milind Padalkar, Ph.D. student in Dhirbhai Ambani Institute of Information and Communication Technology, for his valuable inputs and remarks on several occasions. A special thanks to my colleague Mr. Ainish Dave for his constant help and suggestions throughout my work.

I am also thankful to my brother Saurabh Sohoney for his suggestions and discussions throughout the course of my research work. Last but not the least, I would like to thank my parents for their constant support and encouragement through all these years of study.

Contents

Abstract

In this work, we propose a combined approach to two image processing problems: Image Inpainting and Image Super-Resolution(SR). A number of efficient techniques have been developed for solving these two problems using deep learning, separately. Researchers have developed hierarchical approaches to solve these problems, first in-paint and then super-resolve but there is not much advancement for solving them simultaneously. There are many applications where both inpainting and super-resolution are desired simultaneously like digital reconstruction of invaluable artwork in heritage sites, immersive walk-through systems etc.

We present a supervised learning based approach for simultaneous blind inpainting and super-resolution using Deep Convolutional Neural Network. Network learns mapping between corrupted image patches and true image patches as well as mapping from low resolution features to high resolution features. Trained deep convolutional neural network accepts corrupted low resolution (LR) image as input and outputs a clean high resolution (HR) image. Our network is capable of removing complex patterns from an image and providing higher resolution. However, our focus is limited to simultaneous scratch inpainting and super-resolution.

List of Tables

List of Figures

CHAPTER 1
Introduction

Images get corrupted by random scratches during image acquisition process as the old photographs may get damaged due to cracks. The objective of inpainting algorithms is to recover the original image from noisy observation or to modify an image in a form which is visually pleasing. Missing regions in the image is filled with the help of the inpainting techniques. While image inpainting tries to solve problem of image degradation, image super-resolution is applied to attain an up sampled version of image that enhances image details. Image super-resolution is an algorithmic approach to increase spatial resolution of image.

Although numerous techniques are available for inpainting and SR, methods based on deep learning have achieved better results in both these areas recently and therefore demands more attention. In creating an immersive walk through systems or digital reconstruction of invaluable artwork, primary steps are to inpaint cracks in the damaged region and obtain high resolution. Also, in old damaged photographs, inpainting is useful to recover missing details and SR can be applied for better visual experience. This demands the requirement of simultaneous inpainting and SR.

In this work we present a novel approach for inpainting and super-resolution simultaneously using Convolution Neural Network(CNN). The proposed method is capable of removing random scratches from the images and at the same time provides higher resolution, i.e. we are able to inpaint blindly at higher resolution. We have used a convolutional neural network as used in [3],[4]. It learns an end to end mapping between corrupted low resolution images and corresponding true high resolution images with little pre-processing. This reduces computational complexity and also do not require any prior information about missing region.

1.1 Image Inpainting

Image inpainting is a process to modify an image in a non-detectable form [5]. The aim of inpainting algorithms is to recover missing information in an image (cracks in the old photographs, spots on an image) such that the resultant image is visually plausible. It also includes addition or removal of objects from an image. The missing regions can be filled with the help of the information available in rest of the image. An inpainting technique is not perfect as the information filled is not true, it makes an image complete and visually pleasing. The inpainting problem can be formulated as, given a degraded image and a region inside it which is unknown to us, our task is to fill the missing region. One solution is to modify the pixel values inside the missing region based on the information available outside the region.

Image Inpainting is used in many applications like restoration of images from scratches and superimposed text, removal of objects, providing special effects to images and videos. Inpainting problem can be formulated mathematically and it is an inverse ill posed problem. Priority constraints need to be defined for the solution. Various approaches developed to address the inpainting problem are based on differential equation, patch based, sparsity based, learning based, etc. However, we observed that it is not yet solved accurately. Hence, there exists a scope of exploration in this field. Inpainting can be categorized as non-blind and blind. In non-blind inpainting, the missing region is known in advance to us and an image mask is provided to inpainting algorithm which gives information about the missing pixels, whereas blind inpainting includes automatic detection and correction of damaged region.

1.2 Image Super-resolution

High quality images or videos are desired in most of the imaging applications. Image super-resolution (SR) refers to high spatial resolution or higher pixel density of an image. Increased resolution is desired for improvement of pictorial information for human interpretation. Higher the resolution, better the image detail. Image resolution can be classified into different categories like spatial resolution, temporal resolution, spectral resolution. In this work, we focus on spatial resolution enhancement which is related to pixel density. Spatial resolution refers to the number of pixels required to form a digital image and is measured in pixel per

unit area.

Image super-resolution is very critical in many practical areas like medical imaging, satellite imaging, surveillance etc. Nowadays, HDTVs are popular due to better perpetual quality of video in which image super-resolution is necessary. Super-resolution techniques can also be applied for text enhancement, optical character recognition etc. Development of imaging systems which can capture high spatial resolution images is very expensive. However, high resolution is necessary in certain applications. Therefore, development of super-resolution algorithms is needed to solve this problem. The objective of super-resolution algorithms is to reproduce details at higher resolution.

Image super-resolution techniques can be divided into 2 categories, viz. single frame SR and multi-frame SR.

1.2.1 Single Frame SR

Single frame SR attempts to obtain super-resolution using a single observation of low resolution image. Since available amount of information about scene is less, that makes the problem harder compared to multi frame SR. Exemplar based SR is one such approach in which image patches (examples) are used to obtain super-resolution. Single frame SR approaches super-resolve an image without introducing blur [6].

1.2.2 Multi Frame SR

Multi frame SR is an attempt to generate high resolution(HR) image from multiple low resolution(LR) images of the same scene at sub-pixel alignment. If the observed LR images are at integer pixel shift, then it represents the same information and SR is not possible. But, if enough low resolution images are available at sub-pixel shift,then it is possible to recover the high resolution image. Block schematic for multi frame SR is shown in Figure 1.1

1.3 Deep Learning

Deep learning refers to the set of machine learning techniques which provides solution to many problems in image processing and computer vision. Neural network is used as a deep learning architecture. A deep neural network is a set of stacked layers of artificial neurons. One can learn the parameters of an artificial

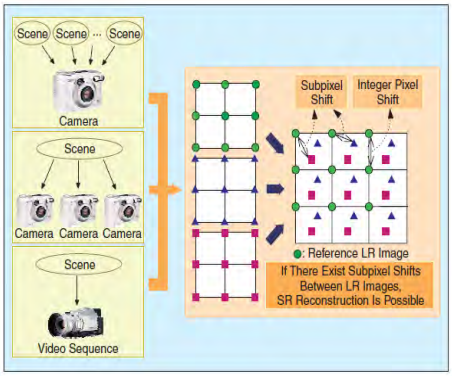


Figure 1.1: Multi frame SR [1]

neural network by training using millions of training examples(in our case images or image patches). Each layer progressively keeps learning low to high level features of the images or image patches, thus learning different levels of abstraction. It learns feature representation of data.

There are three types of leaning algorithms, viz. supervised, semi-supervised and unsupervised. In supervised learning algorithm, hidden layer representations are learned with labeled data, i.e. input and output labels are available at the time of training. In unsupervised learning, network is trained with input data having unlabeled responses. Semi-supervised learning is a combination af supervised and unsupervised learning. At the time of training it makes use of a large amount of unlabeled data with a small amount of labeled data.
Some of the common deep neural network architectures are Multilayer Perceptron (MLP), Autoencoder, Restricted Boltzmann Machine(RBM), Convolutional Neural Network(CNN), Autoencoder, Deep Belief Network(DBN) etc [7].

1.4 Motivation

Inpainting and super-resolution have numerous applications individually and researchers are working on to achieve better results in these areas. Though researchers have approached inpainting and super-resolution in a sequential manner, there is not much progress if these areas are combined. There are many applications where performing inpainting and super-resolution simultaneously is helpful, like digital reconstruction of invaluable artwork, immersive walkthrough systems etc. After doing literature survey, we observed that there is not very significant research done in this direction and there is scope of improvement.

One such issue is finding good models which can handle linear as well as non-linear relationship between input and output. Recent research suggests that non-linear, deep models can give superior results in terms of both speed and accuracy. This is achievable by introducing the concept of deep neural network for learning.

Most of the algorithms developed till now require information about the inpainting region to be given a priori. By training a network for blind inpainting, one can automatically remove complex pattern from the images without any separate processing to detect corrupted regions.

1.5 Problem Definition

Given a low resolution image with scratches, develop an algorithm for simultaneous blind inpainting and super-resolution. Using a set of corrupted low resolution(LR) and high resolution(HR) training images, train a deep neural network to learn mapping from corrupted LR to true HR image and perform super-resolution and inpainitng simultaneously on the given low resolution image.

1.6 Organization of the Thesis

Rest of the thesis is organized as follows. Chapter 2 describes the related work in the areas of inpainting and super-resolution. Chapter 3 describes simultaneous inpainting and super-resolution using stacked denoising autoencoder. In chapter 4, method for simultaneous inpainting and super-resolution is described, which uses deep convolutional neural network. Chapter 5 contains results and their quantitative analysis. Chapter 6 concludes the thesis.

CHAPTER 2
Literature Survey

In the literature survey, we have included some of the previous techniques implemented for inpainting and super-resolution which we have referred to solve our problem.

The term digital image inpainting was first introduced by Bertalmio in [5]. Earlier, image denoising (removal of additive noise, salt and pepper noise) was used in image enhancement applications. In denoising problem, a pixel contains information in addition to small amount of noise. While in inpainting, the pixel information is missing. To address such type of problem, inpainting algorithms were developed.

Image completion techniques are majorly divided in four categories. These comprises of diffusion based methods, exemplar based methods, sparsity based methods and learning based methods. Diffusion based methods are based on propagating information using isotopes lines arriving at boundary of the missing region.The information is smoothly propagated from exterior to interior of the missing region based on solution of partial differential equations(PDE) [5],[8]. The disadvantage with these methods is that they fail to reproduce large textured region.

Second category is exemplar based inpainting. Criminisi et al. in [9] presented a technique for region filling and object removal by exemplar based inpainting. Using this algorithm both texture and structure are propagated in the missing area. The algorithm is based on patch based filling approach. The best matching patch (example) is found from known region and copied to the unknown region. Thus propagating the information. Although, the technique is superior than previously developed approaches in terms of both visual quality and computation efficiency, the limitation is that it fails to obtain reasonable results when similar patches are not found.

Third category is sparsity based inpainting [10, 11]. In this technique, images are represented using a sparse basis (wavelet, DCT etc). Both known and unknown part of the image are assumed to share same sparse representation. This key idea is utilized to recover unknown part of the image using sparse basis. Zongben and Jian have successfully implemented image inpainting using a combination of exemplar based technique and image sparse representation [12]. Idea is that the missing example can be represented with a sparse linear combination of available example.

All the three inpainting techniques discussed above fall under non-blind category. In this case inpainting region should be known in advance, otherwise the algorithm fails. To overcome this drawback, learning based approaches are used which are blind. One such approach is proposed in [2] in which the author uses a 2 layer stacked denoising autoencoder which takes noisy image as an input and outputs a clean image. This is done after training using noisy and corresponding clean images. Kohler et al. in [13] presented an inpainting approach using multi layer perceptron. In their work they have used an image mask in addition to the input noisy image. Third approach corresponds to that of using convolutional neural network similar to our approach [14]. Here, a three layered convolution neural network is used to learn an end to end mapping between a damaged or noisy image and true image. Although all these approaches perform blind inpainting, they can not be applied for the purpose of super-resolution.

While image inpainting recovers information from a degraded image, image super-resolution enhances image details. Image SR has active area of research for the past two decades. A large number of techniques have been developed from frequency domain approach to spatial domain approach and from signal processing to machine learning perspective. Firstly, Tsai and Huang in 1984 proposed a frequency domain approach for multi frame image SR [15]. In this approach, observed LR images are transformed into Discrete Fourier Transform (DFT) domain and combined according to the relationship between aliased DCT of observed LR images and corresponding HR images. This combined data is then transformed back in spatial domain. The new resultant magnified image in spatial domain has the higher spatial resolution than the observed LR images.

Although, frequency domain approaches are less complicated but they are restricted to image observation model, so it can not handle enumerate problems. Therefore, spatial domain SR approaches are applied. Some of the spatial domain

methods for image magnification are interpolation based, iterative back projection, classical multi-image SR, exemplar based SR, sparse coding etc. [16, 17]. Interpolation methods can be used to increase pixel density of an image. But, in these methods image quality is lost because high frequency details are not preserved. Widely used interpolation techniques are bicubic, bilinear, nearest neighbor etc.

One of the spatial domain approaches for single image SR was presented by Yang et al. in [17] using low resolution and high resolution dictionaries. The basic idea is that low resolution and high resolution image pairs share same sparse representation. In this case, we have single image as input and a set of low resolution and high resolution patch pair dictionaries. Our objective is to find the high resolution image using these dictionaries. It is one of the most widely used learning based method which modeled the SR problem as a sparse representation problem. Recently learning based methods have shown promising results for image SR.

Another category of learning based methods uses a trained deep neural network to perform SR. A deep neural network learns a nonlinear mapping from low resolution patches to their corresponding high resolution counterpart. Zhou et al. in [18] trained a deep belief network(DBN) to solve the problem of SR. A DBN is a stacked architecture of multiple RBMs. They used a set of LR and HR images for training. Slightly different from this approach, Nakashika and Ariki proposed a novel technique of SR using DBN [19]. Here, the network is trained with a set of HR images.

Chao Dong et al. [3, 4] introduced the concept of convolutional neural network for SR. In their work, they used a three layer convolutional network that learns the mapping between LR and HR images. They trained the network using bicubic interpolated LR patches as input and original HR patches as output extracted from a large number of images.Their approach showed promisingly good results in terms of speed and accuracy. Motivated from this approach, we have used convolutionl neural network in our work to perform simultaneous inpainting and SR.

After studying the works in the area of inpainting and super-resolution, the need of simultaneous inpainting and super-resolution was observed. Till date, significant amount of work is not available in this area. One of the recent work by Meur et al. gives a hierarchical approach for inpainting and SR[20]. The idea is

to apply inpainting algorithm at coarser resolution and then apply SR algorithms to obtain higher resolution. They have used a combination of exemplar based inpainting and single image SR method. The obtained results were comparable with the existing inpainting and super-resolution techniques but with increased system complexity and time complexity, since two separate methods are applied one by one to get the desired result.

Recently Padalkar et al. performed simultaneous inpainting and super-resolution using self learned dictionaries [21]. The LR and HR dictionaries are created with known region in the test image and its corresponding LR image. However, their approach do not exploit the global characteristics of inpainting regions, since only corrupted LR image is used in performing inpainting and SR. Moreover, the approach is non blind.

CHAPTER 3
Simultaneous Inpainting and SR Using Stacked Sparse Denoising Autoencoder

3.1 Autoecncoder

An auto-encoder neural network is an unsupervised learning architecture that **utilizes** back propagation, setting the target values to be equal to the inputs. It learns the identity function, which is the basic property of an auto-encoder. It represents an architecture for automatic feature learning from unlabeled data. In our work we are using an autoencoder from a different perspective. We train an autoencoder in a supervised manner in which input and output both are available for training. Here input and output both are different but have same dimensions (in our case input is a set of noisy patches and output is set a of clean patches). The network is trained in such a way that it maps input to the output by minimizing error. Note that an autoencoder can also be used for learning, where the hidden layer learns features which are useful in different applications. Figure 3.1 shows an autoencoder with one hidden layer.

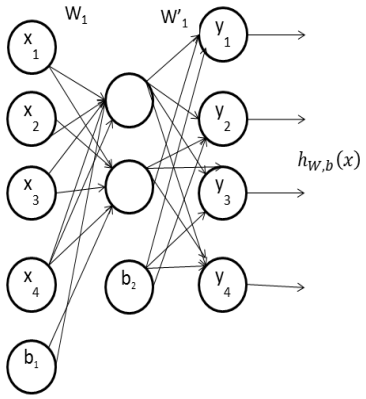


Figure 3.1: Autoencoder

3.2 Stacked Sparse Denoising Autoencoder (SSDA)

A series of more than one cascaded autoencoders are called stacked denoising autoencoder in which the hidden layers features are sparse. Training of SSDA is a greedy learning, as network is first trained layer by layer and then whole network is fine-tuned. If we apply a sparsity constraint on the hidden units, then out of a large number of hidden units most of the hidden units have zero activation and autoencoder is called sparse autoencoder. Under this constraint autoencoder can learn interesting features. Architecture of the stacked denoising auto encoder is shown in figure 3.1. Here W_1, W_2 represents weight matrices and $h_{w,b}(x)$ is network output for input x. An autoencoder is a combination of encoder and decoder therefor weight matrix at layer one is transpose of weight matrix at last layer as shown in figure 3.2.

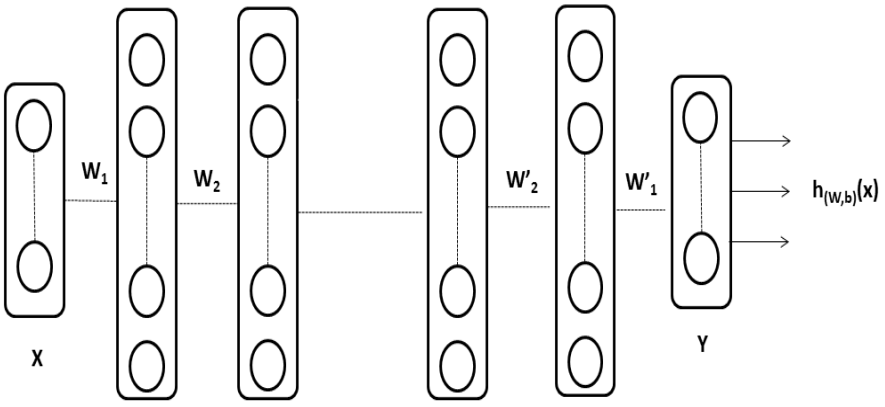


Figure 3.2: Stacked sparse denoising autoencoder [2]

Most of the Inpainting problems can be treated as de-noising problems. Superimposed text or scratches can be modeled as some kind of noise in image. For inpainting purpose the network is trained in a supervised manner in which both the input and output are available for training.

Here, the image Inpainting problem can be formulated as follows. Assume that the scratches on the image represent noise, so the observed noisy image and the original clean image are related by

$$x = \eta(y), \tag{3.1}$$

where η is the image corruption process that corrupts the clean image y to give x. The cost function for obtaining the clean image cab be written as becomes

$$f = argmin_f \|f(x) - y\|^2 \ . \tag{3.2}$$

Now our objective is to find a function f that appropriately reverse the corruption process. In general, network model for inpainting and de-noising problems are dependent on the nature of corruption process η. Researchers have shown results for inpainting a specific kind of structures. For example, the corruption can be Gaussian noise, salt and pepper noise, superimposed text etc. But, in this work we remove scratches from images which do not follow any structure or specific pattern. Since these scratches are random in nature, the task is more challenging.

Let $y^{(i)}, x^{(i)}$ represent for clean and noisy patches/data having N number of pixels in each vector, then the activations of hidden unit and output units for a single input vector in the autoencoder are given by

$$h(x^{(i)}) = \sigma(Wx^{(i)} + b) \quad and \tag{3.3}$$

$$\hat{y}^{(i)} = f(x^{(i)}) = \sigma(W'h(x^{(i)}) + b') \ , \tag{3.4}$$

where, $h(x_i)$ is hidden layer activation, $\hat{y}(i)$ is the network output, σ is the nonlinear sigmoid activation function, given as $\sigma(x) = \frac{1}{1+\exp^{(-x)}}$. From this the system cost function for m training examples can can be given as

$$J(W, b) = \frac{1}{m} \sum_{i=1}^{m} (\|\hat{y}^{(i)} - y^{(i)}\|^2) \ . \tag{3.5}$$

Here m represents the number of patches/data vectors in the training set. To combine sparse coding and avoid over fitting we train a de-noising autoencoder DA to minimize the reconstruction loss regularized by sparsity parameter. The overall cost function is given as,

$$J_{sparse}(W, b) = J(W, b) + \beta KL(\hat{\rho}_j||\rho) + \frac{\lambda}{2}(\|W\|^2 + \|W'\|^2) \ . \tag{3.6}$$

Here, ρ is the sparsity parameter, β controls the weight of sparsity term, λ is the regularization parameter or weight decay parameter, which decrease weights to avoid over fitting. $KL(\hat{\rho}_j||\rho)$ is called Kullback-Leibler divergence. It is a measure of difference between two Bernouli distributions with mean ρ and $\hat{\rho}_j$. $\hat{\rho}_j$ is the average activation of the hidden unit j KL divergence term is given as,

$$KL(\hat{\rho}_j||\rho) = \rho \log \frac{\rho}{\widehat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\widehat{\rho}_j} \ .$$

Our goal is to minimize the overall cost function given in equation 3.6.

After training the first Auto-encoder, hidden layer activations of the trained auto encoder are used for training the second layer. The original noisy training data is passed through the hidden layer and the nonlinear output is used as noisy training data for the next layer. Similarly original clean training data is passed through the hidden layer and the output is used as clean training data for the next layer. Here we have experimented with two stacked de-noising autoencoders. Thus the network has one input layer, 3 hidden layers and one output layer. For training, after a forward pass, the whole network is fine-tuned using back propagation algorithm in order to minimize the final objective function.

3.3 Forward Pass

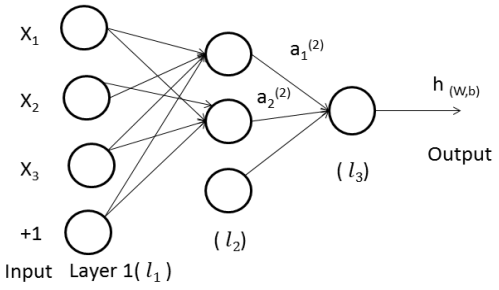


Figure 3.3: Artificial Neural Network

The architecture of a three layer artificial neural network is shown in Figure 3.3. In this network +1 denotes the bias unit and $a_k^{(}l)$ denotes the activation of unit k in layer l. W_{kj} is the weight parameter associated with unit k in layer l and unit j in the layer $l+1$. For the network, the hidden layer activations and the final

network output can be calculated as,

$$a_1^{(2)} = \sigma(W_{11}^1.x_1 + W_{12}^1.x_2 + W_{13}^1.x_3 + b_1) \ , \tag{3.7}$$

$$h_{(W,b)} = \sigma(W_{11}^2.a_1^{(2)} + W_{12}^2.a_2^{(2)} + b_2) \ . \tag{3.8}$$

3.4 Backpropagation

Backpropagation is widely used algorithm for training of various neural network architectures. While training a neural network, firstly a cost function is calculated. A generalized cost function is given as,

$$J(W,b) = \frac{1}{m}\sum_{i=1}^{m}(\|h_{W,b}(x^{(i)}) - y^{(i)}\|^2) + \frac{\lambda}{2}\sum_{l=1}^{l_{last}-1}\sum_{k=1}^{n_l}\sum_{j=1}^{n_{(l+1)}} {W_{(kj)}^l}^2 \ . \tag{3.9}$$

Notations

- l denote the number of layers and l_{last} denotes the last layer.
- k is used as subscript to denote nodes in layer l.
- j is used as subscript to denote nodes in layer $l+1$.
- δ_k^l denotes the error term for k_{th} unit in layer l.
- $f(z_k^l)$ denotes the activation of the node k in layer l which can also be denoted by a_k.
- W_{kj} denotes the weight parameter associated unit j in layer $l+1$ and unit k in layer l.
- m is the number of training examples.

For appropriate training of network, we want to minimize this cost function with respect to W and b. All the weights and biases are first randomly initialized. For each layer l these weights are updated using gradient descent as follows,

$$W^{(l)} = W^{(l)} - \alpha\frac{\delta}{\delta W^{(l)}}J(W,b) \ ,$$

$$b^{(l)} = b^{(l)} - \alpha\frac{\delta}{\delta b^{(l)}}J(W,b) \ ,$$

where α is learning rate. These partial derivatives are calculated using backpropagation algorithm. We give an algorithm to calculate partial derivative with respect to a single training example, then the derivative of the overall cost function can be computed as,

$$\frac{\delta}{\delta W_{kj}^{(l)}} J(W,b) = \frac{1}{m} \sum_{i=1}^{m} J(W,b,x^{(i)},y^{(i)}) + \lambda(W_{kj}^{(l)}) \tag{3.10}$$

$$\frac{\delta}{\delta b^{(l)_k}} J(W,b) = \sum_{i=1}^{m} J(W,b,x^{(i)},y^{(i)}) \tag{3.11}$$

In backpropagation an error term $\delta_{(k)}^{l}$, corresponds to each node in each layer is calculated. This gives a measure of how much that node is contributing in the final error. This error is then backpropagated to input and weights are updated in order to minimize the final error. At the last layer error is given by difference between target value and network output. For hidden unit k this error can be calculate as average sum of the the errors of the nodes to which unit k is connected. The step by step backpropagation algorithm can be given as,

1. Perform a forword pass commputing the activation of the hidden layer and the output layer using equation 3.7 and 3.8.

2. For each node k in the last layer, set

$$\delta_k^l = -(y_k - a_k^{l_{last}}).f'(z_k^{l_{last}})$$

3. For all the hidden layers set,

$$\delta_k^l = (\sum_{k=1}^{n} W_{kj}\delta_k^{(l+1)}).f'(z_k^{(l)})$$

4. **After solving equations 3.10 and 3.11 the partial derivatives are given as,**

$$\Delta_{W(l)} J(W,b,x,y) = a_j^{(l)}.\delta_k^{(l+1)}$$

$$\Delta_{b(l)} J(W,b,x,y) = \delta_k^{(l+1)}$$

The complete algorithm for training a stacked denoising autoencoder is given below.

Algorithm 1: Training of Stacked Denoising Autoencoder

Input : Corrupted LR images/image patched say $x^{(i)}$ / Corresponding true HR images/image patches say $y^{(i)}$

Output: Learned weights and biases $(W_{ih}, b_{ih}, W_{io}, b_{io})$

1. Bi-cubic interpolate corrupted LR images to make input training set say $LR^{'}$
2. Extract millions of patched from LR$^{'}$ as input and from HR as output training data.
3. Initialize a 5 layer SSDA with weights and bias parameters and set $\Delta W^{(l)} = 0, \Delta b^{(l)} = 0$ (Matrix of zeros) for all l, where l is the number of layers.
4. Calculate hidden layer activations and final network output using equations 3.3 and 3.4.
5. Compute the cost function $J_{sparse}(W, b)$ as a function of W and b using equation 3.6.
6. For $i = 1$ to m,
   (a) Use backpropagation to compute $\Delta_{W(l)} J(W, b, x, y)$ and $\Delta_{b(l)} J(W, b, x, y)$.
   (b) Set $\Delta W^{(l)} = \Delta W^{(l)} + \Delta_{W(l)} J(W, b, x, y)$
   (c) Set $\Delta b^{(l)} = \Delta b^{(l)} + \Delta_{b(l)} J(W, b, x, y)$
7. Update the parameter using gradient descent
   (a) $W^{(l)} = W^{(l)} - \alpha[(\frac{1}{m}\Delta W^{(l)}) + \lambda W^{(l)}]$
   (b) $b^{(l)} = b^{(l)} - \alpha[(\frac{1}{m}\Delta b^{(l)}) + \lambda b^{(l)}]$
8. Repeat steps 4 to 7 in order to reduce the cost function.
9. With the learned weights an biases make a forward pass on the test image to get the result using equations 3.3 and 3.4.

3.5 Experimental Setup

We first focus our attention on inpainting grey scale images. We have first experimented for removing random scratches from images. For this we have used a stacked de-noising auto encoder and trained the network with input as scratched patches and output as clean patches. Such 100000 patches of size 7x7 are used for training. We have created our own data set for this experiment by creating random scratches on images. 10 standard natural images of size 128x128 are used from dataset used in [3]. We attempted training of 5 layer network with number of nodes in each layer respectively as [49,245, 245, 245, 49], but due to computational limitations we could get results for single layer autoencoder only.

Next, we extend the above experiment for problem of simultaneous inpainting and SR. Here, the dataset consists of scratched LR images and corresponding true HR image. Same specifications are used as above to create the dataset as well as for network training.

In both the experiments following parameters were set.

1. Weight decay parameter $\lambda = 0.0001$.
2. Weight of sparsity penalty term $\beta = 3$.
3. Desired average activation of the hidden unit $\rho = 0.01$.
4. Hidden size equal to a constant factor times input size. Here we have used that factor $= 5$. Hence the hidden layer has 245 units.
5. Number of iterations m=1000.

Training as well as testing is carried out in MATLAB 2013b.

3.6 Results and Analysis

In this section we present few results for both the experiments i.e, results of inpainting and the results of simultaneous inpainting and SR using SSDA. For quantitative analysis we compare our output with the original bicubic interpolated image in terms of peak signal to noise ratio.

Peak Signal to Noise Ratio (PSNR)

PSNR is the ratio of maximum power of the signal to power of the noise present in the signal. PSNR is usually used to measure the quality of an image after reconstruction. It is an approximation to human perception of quality after reconstruction. A high value of PSNR indicate that signal strength is strong compare to noise and hence quality is better. Therefore, it is used as a measure to compare two images.

To find PSNR, we have to first find difference between two images say A and B of size $m \times n$, which is given by Mean of Square Error (MSE) i.e,

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [A(i,j) - B(i,j)] \quad .$$

The PSNR is then defined as,

$$PSNR = 10 \log_{10} (\frac{MAX^2}{MSE}) \quad .$$

Here MAX is the maximum intensity value of an image.

Blurriness and Blockiness Measures

Farias and Mitra in [22] proposed a method to calculate bulrriness and blockiness of a video by calculating the blurriness and blockiness per frame. We have calculated bluriness and blokiness by considering an image as a frame. Image blurriness is calculated by averaging width over all strong edges in the entire image. If we have K strong edge pixels in an image of size $M \times N$, then the blurriness is given as,

$$blurriness = \frac{1}{K} \sum_{i=0}^{N} \sum_{j=0}^{M} W(i,j) \quad .$$

Here, $W(i,j)$ is width of edge at location (i,j). Width of an edge can be calculated as the distance between two local extremes on each side of the edge.
For calculating blockiness, image is divided into blocks called sub-images and down-sampled in the horizontal and vertical directions. The correlation between the pixels inside and outside the boundaries of the blocks is used as a measure of

blockiness.The correlation between two images S_m and S_n is given as

$$C_{m,n}(i,j) = F^{-1}\frac{F^*(s_m(i,j)).F(s_m(i,j))}{|F^*(s_m(i,j)).F(s_m(i,j))|} \ ,$$

where F denote the Discrete Fourier Transform (DFT) of image, F^{-1} denotes inverse DFT, $*$ denotes the complex conjugate.

(a) Scratched image (b) Inpainted output (c) Original

Figure 3.4: Inpainting on Lena image

(a) Scratched image (b) Inpainted output (c) Original

Figure 3.5: Inpainting on pepper image

(a) Scratched image (b) Inpainted output (c) Original

Figure 3.6: Inpainting on flower image

Figure $3.4 - 3.4$ show the result of inpainting using single layer sparse denoising autoencoder. Inpainted output is shown in (b). First two results are from testing dataset whereas, the third image is from training dataset. From the results we can see that scratches are not completely removed in the images shown in figure $3.4 - 3.6(c)$. Although the network learns inpainting, image quality degrades. The inpainting results can be improved by using more than one layer sparse denoising autoencoder and using larger dataset [2].

Figure $3.7 - 3.9$ show the result of simultaneous scratch inpainting and super-resolution using single layer sparse autoencoder. Looking at the girl image in figure 3.7 and the Monarch image in figure 3.8 we can observe that scratches disappear in the inpainted and super-resolved output in figure $3.7 - 3.8(c)$. However we still find artifacts in the images and can identify these scratch, present in the test image. We observe that from the point of view of super-resolution, image quality is not good.

(a) Scratched LR girl image

(b) Bicubic interpolation of scratched LR

(c) Simultaneous inpainting and SR

(d) Original HR

Figure 3.7: Simultaneous scratch inpainting and SR on girl image

(a) Scratched LR Monarch image

(b) Bicubic interpolation of scratched LR

(c) Simultaneous inpainting and SR

(d) Original HR

Figure 3.8: Simultaneous scratch inpainting and SR on Monarch image

In figure 3.9, we display another result using Barbara image. We can see that as complexity of pattern increases quality of reconstructed image deteriorates. Edges

and image details are lost. From this we can conclude that the network is removing pattern but super-resolution is not achieved. Table 3.1, shows the quantitative comparison of results with the scratched bicubic interpolated LR image and with clean bicubic of original LR image. We can say that PSNR of our result is higher than the PSNR for scratched bicubic LR image but it still lower when compared to the bicubic of original LR image. Table 3.2 shows the assessment of our results in terms of blurriness and blockiness parameters. From these measurements we can say that blurriness and blockiness artifacts are higher in our results when compared to the bicubic interpolation of original LR.

(a) Scratched LR Barbara image

(b) Bicubic interpolation of scratched LR

(c) Simultaneous inpainting and SR

(d) Original HR

Figure 3.9: Simultaneous scratch inpainting and SR on Barbara image

Images	**PSNR for bicubic interpolation of scrached LR**	**PSNR for proposed approach**	**PSNR for bicubic interpolation of original LR**
girl	26.12dB	29.94dB	32.67dB
monarch	20.35dB	27.27dB	32.45dB
barbara	19.71dB	22.45dB	27.99dB

Table 3.1: Comparison of PSNR

Images	**Blurriness for bicubic of original LR**	**Blurriness for simultaneous inpainting and SR**	**Blockiness for bicubic of original LR**	**Blockiness for simultaneous inpainting and SR**
girl	6.4	6.51	0.0855	0.1581
Monarch	5.34	5.9	0.0054	0.0867
Barbara	5.89	7.81	0.2214	0.1475

Table 3.2: Comparison of blurriness and blockiness

3.7 Autoencoder to Convolutional Neural Network

In the previous section we presented the results for simultaneous inpainting and super-resolution using stacked sparse denoising autoencoder. The results shown are for non blind approach i.e. information about the inpaining region is provided at the time of training as well a at the time of testing, therefore objective of blind inpainting is not achieved. Also, we can see from the results that image quality is not preserved while performing super-resolution.

The reason for no improvement in performance is due to inability of autoencoder to capture local correlations in image. Autoencoder is used specifically to learn compact, data specific representation. It is not a good architecture for learning a mapping from low resolution to high resolution features. Also while training and autoencoder large number of parameters are required to be trained, which leads to ver fitting. These problems can be solved using convolutional neural network, which uses convolution operations to capture local correlations. Convolutional neural network can learn mapping from low resolution images to high resolution images [3]. Therefore in next chapter we used convolutional neural network for achieving better results.

CHAPTER 4
Simultaneous Blind Inpainting and SR Using Convolutional Neural Network

4.1 Convolutional Neural Network (CNN)

CNN represents an artificial neural network architecture designed for classification and visual recognition [13,14]. CNN is similar to other neural network architectures and has learnable weights and biases but the objective of CNN is to learn data specific filters. Earlier it was used only for classification purpose where the output of network is binary or in case of multi-class classification problems it can only predict class to which the object belongs to. In recent years CNN gained popularity and is used successfully for solving reconstruction problems as well, where it outputs an entire image [3],[4],[14]. In our work we show that CNN takes low resolution image having scratches as input and using the learned weights and biases, outputs an image that closely resembles to true high resolution image. Fig-

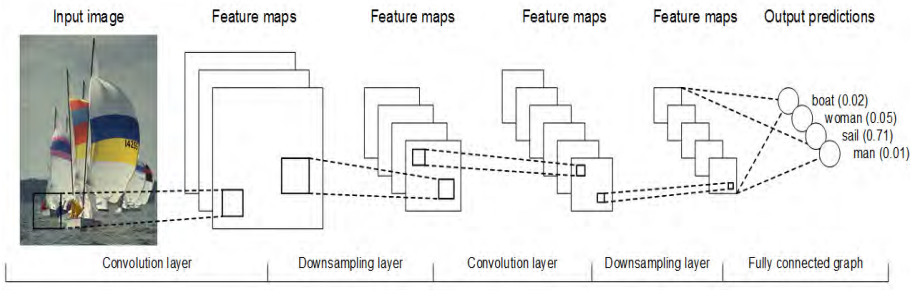


Figure 4.1: Basic CNN block diagram

ure 1. shows the architecture of a basic CNN used for classification. There are few well defined layers in a CNN such as convolutional layer, pooling layer and fully connected (FC) layer. Here unlike sigmoid function a rectified linear unit (ReLU) is used for non-linear mapping. These are defined as,

Convolution Layer

Convolution layer simply performs the operation of image convolution on the input images. Here kernel is specified by weight matrix. The kernel is connected to local region in the previous layer or input and slides over entire image. The output of the convolution layer is given as a set of neurons which are computed as dot product of weight matrix and local region to which they are connected.

Rectified Linear Unit (ReLU)

The output of convolution layer is nonlinearly mapped using ReLU activation function, which is given as $f(x) = max(0, x)$. Functioning of RelU is similar to a rectifier, as it maps all the negative values to zero and keeps all the positive values. There are many advantages of using ReLU over sigmoid or tanh functions. It results in faster and efficient training of the network. ReLU also introduces sparsity in the network as most of the hidden units are zero, which results in effective feature learning.

Pooling Layer

In the basic CNN or CNN which are used for classification purpose, pooling layer is used after convolution layer. This is used for feature dimensionality reduction. Pooling layer takes a block as input from the convolutional layer, sabsample it and produces a single output. There are various methods for performing this polling operation such as max pooling, min pooing, average pooling etc. In most of the CNN architectures max pooling is used in which maximum value of the block is produced as the output.

Fully Connected Layer

Fully connected layer is used as the last layer in CNN. It specifies object class. If size of the fully connected layer is $[1 \times 1 \times n]$, then index n specifies the number of classes to which the object belongs.

The training of CNN leads to hierarchical feature learning i.e, low level features to high level features. Network learns the parameter weight matrix W and bias vector b in each layer. Functioning of different layers is a combination of linear

filtering using convolution and nonlinear mapping using ReLU. Hidden layers learn features which map input to output after training.

4.2 Proposed Approach for Simultaneous Blind Inpainting and SR

In this section we give problem formulation and description of CNN used. Then we briefly give algorithm for training and testing.

Problem Formulation :

Assume that X is a corrupted low resolution image and Y is the corresponding high resolution image. In order to make them to have same dimensions, we first upsample X by bicubic interpolation to obtain $X^{'}$. Now our task is to recover image Z form $X^{'}$ that closely resembles the ground truth image Y. We can formulate the problem mathematically as

$$Z = f(X^{'}) \quad . \tag{4.1}$$

To do this we train a network structure that learns the mapping function f which minimizes the error between Y and Z. In our work we use both Y and X to learn mapping f. Note that we have combined the problem of inpainting and super-resolution as the mapping function f learns to remove degradation in addition to learning the high resolution details.

4.2.1 Deep CNN Architecture for Simultaneous Blind Inpainting and SR

Inspired from the work in [3],[4], we have also used a 3 layer convolutional neural network for simultaneous inpainting and SR. All the three layers are convolution layers. Pooling layer and fully connected layer are not used in reconstruction problems because here we do not want dimensionality reduction and class scores, as done in classification. Figure 2 shows the complete architecture of deep CNN used in our work.

In the first step, training images are represented by a large number of sub-images. To process an image one should represent these images in the form of

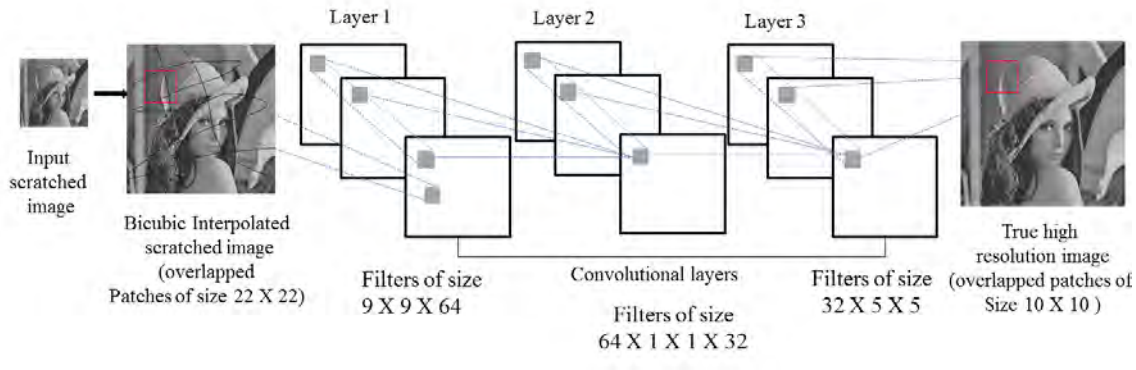


Figure 4.2: Deep CNN architecture

predefined basis which is equivalent to convolution representation of image. Convolutionl layer computes the output of neurons that are connected to a local region at the input by convolving an image using a set of filters. Each filter computes a dot product between their weights and region they are connected at the input. In our network we have a set of 3 cascaded convolutional layers of different sizes.

Let layer l be number of convolutional layers, the output of convolution at layer l is mathematically expressed as

$$Y_l = W_l * h_{(l-1)} + B_l \ , \tag{4.2}$$

where B is the bias matrix, W is the weight matrix that is set of filters or feature maps, $h_{(l-1)}$ is activation of previous hidden layer and $*$ is a convolution operator. Here W_1 is of size $k_1 \times k_1 \times n_1$ as shown in figure, where k_1 and n_1 represent the size and number of filters, respectively at layer 1 . For $l = 1$, h_0 corresponds to the input X. To find hidden layer activation, convolved output is non-linearly mapped by the transformation given by

$$F(y) = max(0, y) \ . \tag{4.3}$$

ReLU is used in convolutional neural network for faster convergence of stochastic gradient descent compared to sigmoid/tanh functions [18]. ReLU can be implemented by thresholding matrix of activations to zero. In the final step the overlapped patches are averaged to get the final output image. This averaging is performed at last convolutional layer output of which is given by,

$$F_l(y) = W_l * F_{l-1}(y) + B_l \quad . \tag{4.4}$$

Here we give an algorithm for training of our network.

Algorithm 2: Training of deep CNN

Input : Corrupted LR images say $x^{(i)}$ / Corresponding true HR images say $y^{(i)}$

Output: Learned weights and biases $(W_1, W_2, ...W_l, b_1, b_2.....b_l)$

1. Bi-cubic interpolate corrupted LR images to make input training set say $LR^{'}$
2. Extract millions of sub-images from LR$^{'}$ as input and from HR as output training data.
3. Initialize a 3 layer CNN with weights and bias parameters.
4. Calculate hidden layer activations and final network output using equation $(4.2), (4.3)$ and (4.4).
5. Compute the cost function

$$J(W, b) = \frac{1}{m} \sum_{i=1}^{m} (\|h_{W,b}(x^{(i)}) - y^{(i)}\|^2) \quad ,$$

   where $h_{W,b}(x^{(i)}$ is deep CNN output for input sub-image $x^{(i)}$ and m is number of samples.
6. Optimize the cost function by updating the parameters using stochastic gradient descent and standard back-propagation.
7. With learned weights and biases perform a forward pass on test image using equation $(4.2), (4.3)$ and (4.4) to get the desired output.

4.3 Experimental Setup

In order to test the performance of our approach we performed experiments on natural images. Since no dataset is available for simultaneous inpainting and super-resolution, we created our own data set using datasets used in [9]. This dataset consists of 91 natural images which are used to make corrupt LR set X as input and true HR set Y as output. Set 5 of [3] is used to evaluate performance of upsampling factor 2. In order to make corrupt LR images we down-sampled and manually corrupted true HR images using photo-shop. Note that scratches on corrupted LR images in training data as well as scratched on testing LR images do not follow any specific kind of pattern i.e it is random in nature. To construct input data set, these scracthed LR images are upsampled using bicubic interpolation.

4.3.1 Implementation Details

For each corrupt pattern and up-sampling factor a specific deep CNN is trained. We trained our network for simultaneous scratch removal and super-resolution. To train a network for a factor of 2, 90000 sub-images are extracted from input and output training data sets. Network needs to be trained separately for different upscaling factors. Here sub-images means large sized patches. Overlapping sub-images are extracted from original images with a stride i.e. pixel distance of 4. By training network with overlapping sub-images, we capture the local correlations. The weights for filters are initialized randomly using samples from a Gaussian distribution with zero mean and variance of 0.001. Let K_l and n_l denote the size of filter and number of filter in layer l respectively. We have used a three layer CNN with $k_1 = 9$, $k_2 = 1$ $k_3 = 5$, $n_1 = 64$, $n_2 = 32$ i.e, 64 filters of size 9×9 is used at layer 1, 32 filters of size 1×1 is used at layer 2 and single filter of size 5×5 is used at the last layer. Other network parameters are chosen as,

- Learning rate for output layer $\alpha = 5 \times 10^{-3}$.
- Learning rate for the other layers $\alpha = 5 \times 10^{-5}$.
- Momentum $M = 0.9$.

Momentum is used for accelerating optimization and to quickly reach the local minimum. Smaller learning rate at last layer is required for convergence [9].To avoid border effects during training, we avoid using zero padding. Due to this network output has smaller size when compared to input . Here we used input

sub-image of size 22×22 and output sub-image has a size of 10×10. MSE is calculated between the center 10×10 cropped input and output 10×10 subimage. However we want that network should be applied to arbitrary images of different size. Therefore sufficient amount of zero padding is provided during testing which results in same output dimensions as input. We have experimented only on luminance channel (YCbCr) for training and testing.

We used Caffe with CPU to train our network. Caffe is a deep learning library in which network models and optimizations are defined. It was developed by Berkeley vision and learning center. Various fuctions are built in C++ and Python. Training can be done on Central Processing Unit (CPU) as well as Graphical Processing Unit (GPU). All the experiments are performed on Intel i_7 processor with 16 GB RAM. It takes 5 days to complete 10 lakhs iterations on this CPU. In Caffe implementation is done using cuda-convnet package [3].

CHAPTER 5
Results and Analysis

Feature learning of CNN used for reconstruction problem is similar to feature learning of the CNN used for classification,. Filters learned in each layer show the hierarchical nature of features in the network. In the network used, layer 1 corresponds to the edges in the images. Layer 2 corresponds to corners and other edge/color conjuctions and layer 3 has more features capturing mesh, patterns etc.[23]. In terms of frequency details, one can say that first layer filters are a mix of extremely low and high frequency components with a little converge of mid frequency components. Last layer filters are high frequency components. In this way low to high level features are learned in the convolutional neural network.

In this chapter, we present the results of our experiment performed on natural images. Due to computational limitations we show results by training the network for 5000000 iterations as opposed to 1,5000000 iterations performed by Dong et al. [3]. We show comparative results using the peak signal to noise ratio(PSNR), blurriness and blockiness [22]. To the best of our knowledge there are no existing methods for simultaneous blind scratch inpainting and SR. We, therefore compare our results with bicubically interpolated version of original LR image. Figure 5.1-5.6 show the results of simultaneous blind scratch inpainting and super-resolution for a factor of 2 in . In each figure, (a) shows the scratched LR image, (b) shows its bicubically interpolated version. The simultaneously inpainted and super-resolved output using proposed approach is shown in (c). The bicubically interpolated version of the true LR image and the original HR image are shown in figures (d) and (e), respectively. Figures (f) and (g) shows the enlarged version of the regions marked in figures (c) and (d), respectively.

Figure 5.1 shows the simultaneous blind inpainting and SR on boy image. From the result in figure 5.1(c), we can clearly see that scratches are removed without affecting the image quality. From the figures 5.1(d) and 5.1(e), we can see that

texture near the eyes, nose and hair are better present in our result, as the result looks similar to bicubic interpolation of true LR. This indicates that our approach removes the scratches as well as it upsamples the image with the details being preserved.

In order to evaluate the performance of this algorithm on a more complex image consisting of a cluttered scene, we shown the result on the bird image in figure 5.2. Observe that in even in such a complex image containing a cluttered scene, the scratches have been removed. At the same time the simultaneously inpainted and super-resolved image shown in $5.1(c)$ also retains various details which are not visible in bicubically interpolated version of true LR image shown in $5.1(d)$.

Figure 5.3 shows simultaneous inpainting and super-resolution on the scratched Lena image. One may note that this is a more challenging case as the image contains higher number of edge features. This is because the algorithm not only needs to maintain the connectivity of these edges but also their sharpness across the inpainted pixels. Notice the inpainted area of the left eye. Using the proposed approach, the inpainting is not only is seamless but also retains the edge details inside the eye. Moreover, the various edges in the hat are clearly more sharper than those in the bicubically interpolated version of the true LR image.Yet another result for scratch inpainting and SR on an image containing cluttered objects is shown using the pepper image in figure 5.4. From this figure we can observe that image texture is well preserved for each object in the result of our approach.

In order to show that our trained model can distinguish between black scratches and edges or lines present in an image we consider the Monarch image shown in figure 5.5. From the inpainted and super-resolved output in $5.5(c)$, we can see that only scratches are removed while other details that appear similar to scratches (due to the presence of dark edges) have been successfully retained. This result is a good example for showing effectiveness of our approach for super-resolution in addition to blind inpainting because the minute details of both the butterfly and the flower-petals have been enhanced and these look visually similar to the ground truth shown in figure 5.5(e). Similarly, consider the results shown in figure 5.6 of a home image which also consists of large number of edge features. Even in this case, the proposed algorithm provides a seamless inpainting with sharper edge details.

The comparison between the PSNR values of results using (a) bicubically interpo-

lated version of the scratched LR image, (b) our proposed approach and (c) bicubically interpolated version of the original LR image are shown in table 5.1. Here, we observe that in each case the PSNR for our result is significantly higher than that for the bicubically interpolated version of the scratched LR image. Also the PSNR values of our results are almost comparable to those for the bicubically interpolated versions of the original LR images. Note that, due to the limitations on the available computational resources, we could train our model for only 5000000 iterations, which is a significantly small number compared to the number of iterations performed by the state-of-the-art model [3]. The objective quality of the results using our method in terms of PSNR can be improved if the model is trained using higher number of iterations. Note that, with our present trained model the PSNR is greater than that for bicubically interpolated version of the original LR image in the case of the Monarch image shown in figure 5.5.

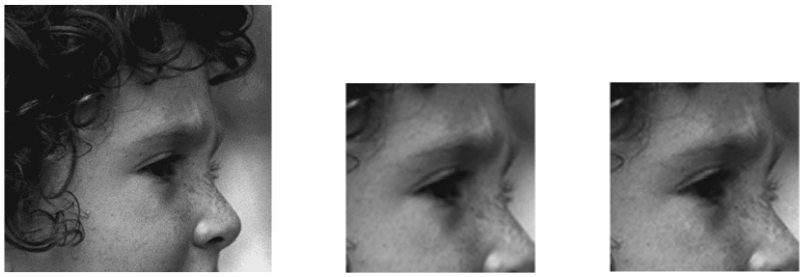
Table 5.2 shows comparison of our result with that of bicubic interpolation of original LR image in terms of the blurriness and blockiness quality metrics. Lower the values of these metrics, better is the image quality. For all the images, we can observe that for our results, the blurriness and blockiness matrices have smaller values in comparison to the interpolated version of the original LR image. Hence we can say that the proposed method performs simultaneous blind inpainting and super-resolution while avoiding the blurriness and blockiness artifacts.

(a) Scratched LR image

(b) Bicubically interpolated scratched LR

(c) Simultaneously inpainted and supre-resoved image

(d) Bicubic interpolation of true LR

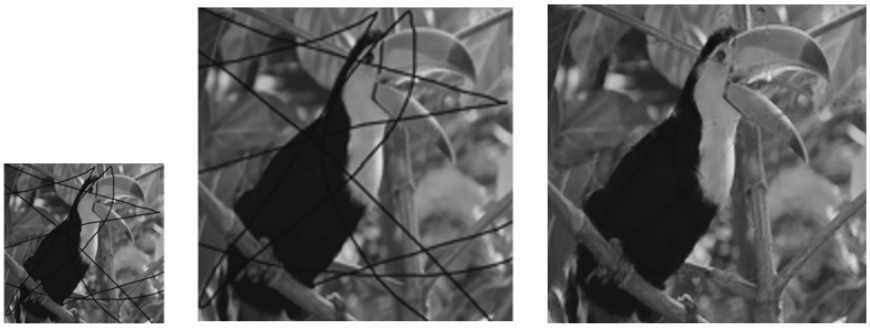


(e) Original HR

(f) Enlarged image of the region shown in (c)

(g) Enlarged image of the region shown in (d)

Figure 5.1: Simultaneous scratch inpainting and SR on boy image



(a) Scratched LR image

(b) Bicubically interpolated scratched LR

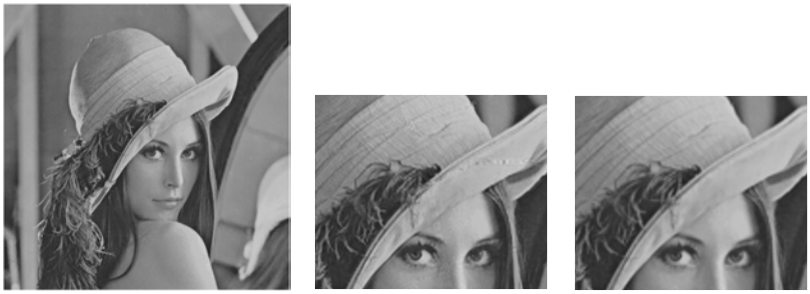
(c) Simultaneously inpainted super-resolved image

(d) Bicubic interpolation of true LR

(e) Original HR

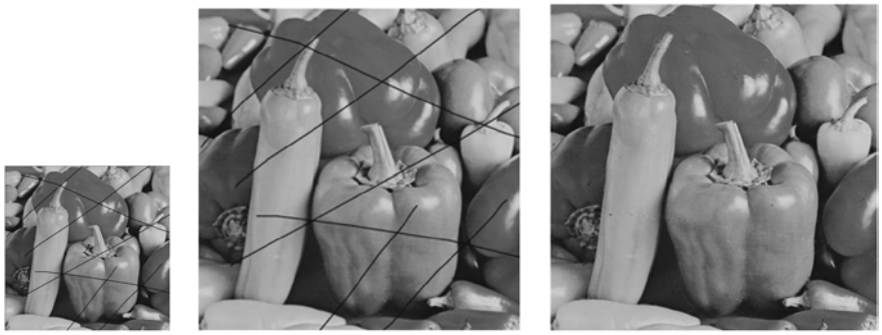
Figure 5.2: Simultaneous inpainting and SR on bird image

(a) Scratched LR image (b) Bicubically interpolated scratched LR (c) Simultaneously inapainted and super-resolved image (d) bicubic interpolation of true LR



(e) Original HR (f) Enlarged image of the region shown in (c) (g) Enlarged image of the region shown in (d)

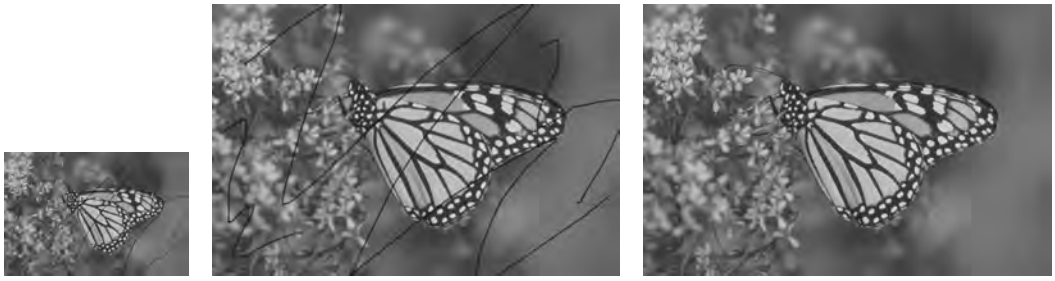
Figure 5.3: Simultaneous scratch inpainting and SR on Lena image



(a) Scratched LR (b) bicubically interpolated scratched LR (c) Simultaneously inpainted and super-resolved image

(d) bicubic interpolation of true LR (e) Original HR

Figure 5.4: Simultaneous inpainting and SR on pepper image



(a) Scratched LR image

(b) Bicubically interpolated scratched LR

(c) Simultaneously inpainted and super-resolved image

(d) bicubic interpolation of true LR

(e) Original HR

Figure 5.5: Simultaneous inpainting and SR on Monarch image

(a) Scratched LR image

(b) Bicubically interpolated scratched LR

(c) Simultaneously inpainted and super-resolved output

(d) Bicubic interpolation of true LR

(e) Original HR

(f) Enlarged image of the region shown in (c)

(g) Enlarged image of the region shown in (d)

Figure 5.6: Simultaneous scratch inpainting and SR on home image

Images	**PSNR for bicubic interpolation of scrached LR**	**PSNR for proposed approach**	**PSNR for bicubic interpolation of original LR**
boy	22.18dB	33.61dB	34.85dB
bird	24.43dB	33.72dB	35.80dB
Lena	24.67dB	34.13dB	34.69dB
pepper	25.48dB	34.85dB	34.95dB
Monarch	26.57dB	33.33dB	32.95dB
home	18.33dB	31.34dB	32.89dB

Table 5.1: Comparision of PSNR

Images	**Blurriness for bicubic of original LR**	**Blurriness for simultaneous inpainting and SR**	**Blockiness for bicubic of original LR**	**Blockiness for simultaneous inpainting and SR**
boy	5.4138	5.2584	0.2718	0.1250
bird	5.0049	5.09	0.2718	0.1250
Lena	5.8324	5.6216	0.2404	0.1433
pepper	6.3044	6.1884	0.1935	0.0980
Monarch	5.34	5.27	0.2105	0.0030
home	5.1003	4.979	0.1305	0.0836

Table 5.2: Comparison of image blurriness and blockiness

CHAPTER 6
Conclusion

In this work, we have presented a new deep learning based approach for simultaneous blind inpainting and super-resolution. The idea is to train a deep neural network with a large number of scratched low resolution and corresponding true high resolution image pairs. Initially, we used a stacked sparse denoising autoencoder to solve the problem simultaneous blind inpainting and super-resolution. However, it was observed that, there are problems while training an autoencoder, which can be solved with the help of other neural network architecture.

To overcome the drawbacks of an autoencoder in performing simultaneous inpainting and super-resolution, a deep convolution neural network(CNN) is used. The proposed network is blind, i.e. it fills the missing pixels or remove complex pattern from images without manually masking them prior to inpainting. The proposed approach of simultaneous inpainting and super-resolution using CNN is supported by the results for images with varying complexity. The obtained results show that random scratches are completely removed and spatial resolution is increased. Moreover, there is no loss of quality in the resultant image.

Although a specific network needs to be trained for a particular kind of noise (scratches, super-imposed text), the network architecture remains same. The network performance can further be increased by increasing number of layers in CNN. Using a different set of training image pairs, the network can also be applied for super-resolution with higher upsampling factors.

References

[1] Sung Cheol Park, Min Kyu Park, and Moon Gi Kang. Super-resolution image reconstruction: a technical overview. *Signal Processing Magazine, IEEE*, 20(3):21–36, 2003.

[2] Junyuan Xie, Linli Xu, and Enhong Chen. Image denoising and inpainting with deep neural networks. In *Advances in Neural Information Processing Systems*, pages 341–349, 2012.

[3] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *CoRR*, abs/1501.00092, 2015.

[4] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part IV*, chapter Learning a Deep Convolutional Network for Image Super-Resolution, pages 184–199. Springer International Publishing, Cham, 2014.

[5] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 417–424, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[6] Jianchao Yang and Thomas Huang. Image super-resolution: Historical overview and future challenges. *Super-resolution imaging*, pages 1–34, 2010.

[7] Joost van Doorn. Analysis of deep convolutional neural network architectures. 2014.

[8] Joachim Weickert. *Theoretical foundations of anisotropic diffusion in image processing*. Springer, 1996.

[9] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. Region filling and object removal by exemplar-based image inpainting. *Image Processing, IEEE Transactions on*, 13(9):1200–1212, 2004.

[10] Zongben Xu and Jian Sun. Image inpainting by patch propagation using patch sparsity. *Image Processing, IEEE Transactions on*, 19(5):1153–1165, 2010.

[11] Bin Shen, Wei Hu, Yimin Zhang, and Yu-Jin Zhang. Image inpainting via sparse representation. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 697–700. IEEE, 2009.

[12] Zongben Xu and Jian Sun. Image inpainting by patch propagation using patch sparsity. *Image Processing, IEEE Transactions on*, 19(5):1153–1165, 2010.

[13] Rolf Köhler, Christian Schuler, Bernhard Schölkopf, and Stefan Harmeling. Mask-specific inpainting with deep neural networks. In *Pattern Recognition*, pages 523–534. Springer, 2014.

[14] Nian Cai, Zhenghang Su, Zhineng Lin, Han Wang, Zhijing Yang, and Bingo Wing-Kuen Ling. Blind inpainting using the fully convolutional neural network. *The Visual Computer*, pages 1–13, 2015.

[15] T. S. Huang R. Y. Tsai. Multiframe image restoration and registration. In *Advances in Computer Vision and Image Processsing*, pages 317–339, 1984.

[16] Robinson Macwan, Nehal Patel, Priteshkumar Prajapati, and Jaimin Chavda. A survey on various techniques of super resolution imaging. *International Journal of Computer Applications*, 90(1), 2014.

[17] Jianchao Yang, John Wright, Thomas S Huang, and Yi Ma. Image super-resolution via sparse representation. *Image Processing, IEEE Transactions on*, 19(11):2861–2873, 2010.

[18] Yanwen Zhou, Yanyun Qu, Yuan Xie, and Wensheng Zhang. Image super-resolution using deep belief networks. In *Proceedings of International Conference on Internet Multimedia Computing and Service*, page 28. ACM, 2014.

[19] Toru Nakashika, Tetsuya Takiguchi, and Yasuo Ariki. High-frequency restoration using deep belief nets for super-resolution. In *Signal-Image Technology & Internet-Based Systems (SITIS), 2013 International Conference on*, pages 38–42. IEEE, 2013.

[20] Olivier Le Meur, Mounira Ebdelli, and Christine Guillemot. Hierarchical super-resolution-based inpainting. *Image Processing, IEEE Transactions on*, 22(10):3779–3790, 2013.

[21] Milind G. Padalkar, Manjunath V. Joshi, and Nilay Khatri. Simultaneous inpainting and super-resolution using self-learning. In Mark W. Jones Xianghua Xie and Gary K. L. Tam, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 105.1–105.12. BMVA Press, September 2015.

[22] Mylene CQ Farias and Sanjit K Mitra. No-reference video quality metric based on artifact measurements. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 3, pages III–141. IEEE, 2005.

[23] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer vision–ECCV 2014*, pages 818–833. Springer, 2014.