Project Report on

A STUDY ON IMAGE SUPERRESOLUTION FROM MULTIPLE FRAMES USING ARTIFICIAL NEURAL NETWORK

Submitted to University Grants Commission, Bahadurshah Marg, New Delhi

In respect to the Minor Project Sanctioned

No. F.MRP/12th Plan/14-15/KLMG073 dated 10.12.14

Project Period 27/01/2015 to 31/03/2017

Submitted By

Dr. Saritha M,
Associate Professor,
Department of Electronics,
NSS College, Rajakumari,
Idukki, Kerala

ABSTRACT

Super resolution imaging refers to inferring the missing high resolution image from low resolution image(s). Super resolution methods are generally classified into reconstruction based and learning based methods. In this project a neural network is used for the super resolution. Neural networks are among the most important machine learning techniques and thus good candidates for a project in artificial intelligence. A neural network is a data processing system consisting of a large number of simple, highly interconnected processing elements in an architecture inspired by the structure of the cerebral cortex portion of the brain. Hence, neural networks are often capable of doing things which humans or animals do well but which conventional computers often do poorly. They have emerged in the past few years as an area of unusual opportunity for research, development and application to a variety of real world problems. Indeed, neural networks exhibit characteristics and capabilities not provided by any other technology. In this project first we train the network using low resolution images and high resolution images, and then we give another low resolution image to the network. It will produce high resolution image of the low resolution image. Neural networking applications are very fast and will give good results when comparing with the normal methods.

1. Introduction

Image Super-Resolution is the most widely and expensive area of research and they can decade to solve the problem of limited resolution by image acquisition devices and also dependent on sensor. But, high- resolution sensor is very expensive. And for image acquisition could be as simple and it also involves preprocessing such as scaling. Also the available camera may not always sufficient for any given application. So, we need increase the current resolution by two ways, either reducing the pixel size or by increasing the chip size. However it has some limitations which can generate noise and degrade the image quality. Therefore, a new method is required to increase the resolution of the image. Super resolution can used many application likes Medical imaging, Satellite imaging, Remote imaging, Video surveillance, Enlarging consumer photograph for higher quality, Enlarging consumer photograph for higher quality. Now, Super-Resolution (SR) is to obtain a high-resolution (HR) image using one or more observed low-resolution (LR) images by down-sampling, de-blurring, and de-noising. Where, Low Resolution (LR) image represents low pixel quality and it provide less accurate details. High-Resolution (HR) represents high pixel quality and it provides more accurate details.

SR techniques can prove useful in many different applications, and these applications can have different requirements in terms of both quality and computational complexity. The quality may also vary for different methods based on characteristics of the input image. The implementation complexity may be affected by implementation specifics, such as the availability of specific optimized libraries. Finally the artifacts caused by poor SR performance can be more visually distracting than blurring from interpolation. For these and other reasons choosing between SR methods is a complex task.

1.1 Super Resolution

Super-resolution, loosely speaking, is the process of recovering a high-resolution image from a set of low resolution input images or from a single low resolution image. Any given set of source low resolution (LR) images only captures a finite amount of information from a scene; the goal of SR is to extract the independent information from each image in that set and combine the information into a single high resolution (HR) image. The only requirement is that each LR image must contain some information that is unique to that image. This means that when these LR images are mapped onto a common reference plane their samples must be sub pixel shifted from samples of other images – otherwise the images would contain only redundant information and SR reconstruction would not be possible.

In the Super-resolution techniques, that classifies into two major parts: Frequency domain approach, and spatial domain approach. Frequency domain approach, which can perform Fourier transform of an image. These methods are simple and computationally cheap, they are extremely sensitive to model error, limiting their use and Spatial domain approach, which can perform directly on pixel and it is also more popular method. These methods are computationally expensive.

1.1.1 Frequency Domain Methods

A major class of SR methods utilizes a frequency domain formulation of the SR problem. Frequency domain methods are based on three fundamental principles:

i) The shifting property of the Fourier transform (FT)

ii) The aliasing relationship between the continuous Fourier transform (CFT) and the discrete Fourier transform (DFT)

iii) The original scene is band-limited.

These properties allow the formulation of a system of equations relating the aliased DFT coefficients of the observed images to samples of the CFT of the unknown scene. These equations are solved yielding the frequency domain coefficients of the original scene, which may then be recovered by inverse DFT. Formulation of the system of equations requires knowledge of the translational motion between frames to sub-pixel accuracy. Each observation image must contribute independent equations, which places restrictions on the inter-frame motion that contributes useful data

1.1.2 Spatial Domain Methods

Approaching the super-resolution problem in the frequency domain makes a lot of sense because it is relatively simple and computationally efficient. However, there are some problems with a frequency domain formulation. For one, it restricts the inter-frame motion to be translational because the DFT assumes uniformly spaced samples. Another disadvantage is that prior knowledge that might be used to constrain or regularize the super-resolution problem is often difficult to express in the frequency domain. Since the super-resolution problem is fundamentally ill-posed4, incorporation of prior knowledge is essential to achieve good results. A variety of techniques exist for the super-resolution problem in the spatial domain. These solutions include interpolation, deterministic regularized techniques, stochastic methods, iterative back projection, and projection onto convex sets among others. The primary advantages to working in the spatial domain are support for unconstrained motion between frames and ease of incorporating prior knowledge into the solution.

1.1.3 Implementation of Super-Resolution

For the technical implementation of Super-Resolution in two ways: Single-frame and Multi-frame image Super Resolution. Single-frame Super-Resolution methods to generate single high-resolution image from single degraded or noisy or blurred image. And Multi-frame Super-Resolution is to generate the high-resolution (HR) image from multiple low-resolution images perspectives of a same scene and also increase spatial resolution by fusing information.

There are mainly three methods for this implementation: interpolation based methods, reconstruction based methods, and learning based methods. The interpolation based methods are simple but tend to blur the high frequency details. The reconstruction based methods enforce a reconstruction constraint which requires that the smoothed and down-sampled version of the HR image should be close to the LR image. The learning based methods "hallucinate" high frequency details from a training set of HR/LR image pairs. The learning based approach highly relies on the similarity between the training set and the test set. It is still unclear how many training examples are sufficient for the generic images.

1.2 Methodology

After an extensive study of image resolutions, we have decided to use the following technique to develop a robust, credible super resolution system

- Neural network
- Back propagation algorithm

A neural network is a computer system modeled on the human brain and nervous system and Back propagation algorithm is the algorithm used for teaching the neural network.

The super resolution system operates on two phases. The first phase is involves finding the weights of the neural network. This process is called training or teaching. Second phase is the creation of high resolution image using this trained neural network.

1.2.1 Neural Network

Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the connections between elements largely determine the network function. We can train a neural network to perform a particular function by adjusting the values of the connections (weights) between elements.

Typically, neural networks are adjusted, or trained, so that a particular input leads to a specific target output. The next figure illustrates such a situation. There, the network is adjusted, based on a comparison of the output and the target, until the network output matches the target. Typically, many such input/target pairs are needed to train a network

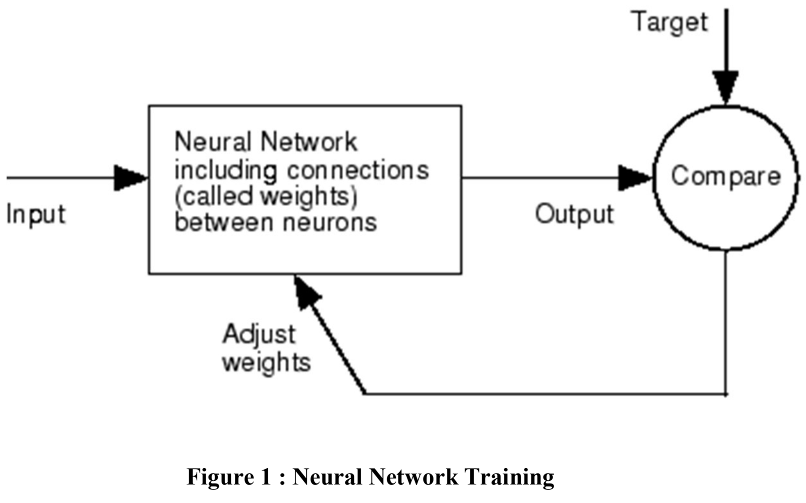

Target
Neural Network including connections (called weights) between neurons
Input
Output
Compare
Adjust weights

Figure 1 : Neural Network Training

Neural networks have been trained to perform complex functions in various fields, including pattern recognition, identification, classification, speech, vision, and control systems. Neural networks can also be trained to solve problems that are difficult for conventional computers or human beings.

1.2.2 Back Propagation Algorithm

Back Propagation, an abbreviation for "backward propagation of errors", is a common method of training artificial neural networks used in conjunction with an optimization such as gradient descent. The method calculates the gradient of a loss function with respect to all the weights in the network. The gradient is fed to the optimization method which in turn uses it to update the weights, in an attempt to minimize the loss function.

Back propagation requires a known, desired output for each input value in order to calculate the loss function gradient. It is therefore usually considered to be a supervised learning method, although it is also used in some unsupervised networks such as auto encoders. It is a generalization of the delta rule to multi-layered feed forward networks, made possible by using the chain rule to iteratively compute gradients for each layer. Back propagation requires that the activation function used by the artificial neurons (or "nodes") be differentiable.

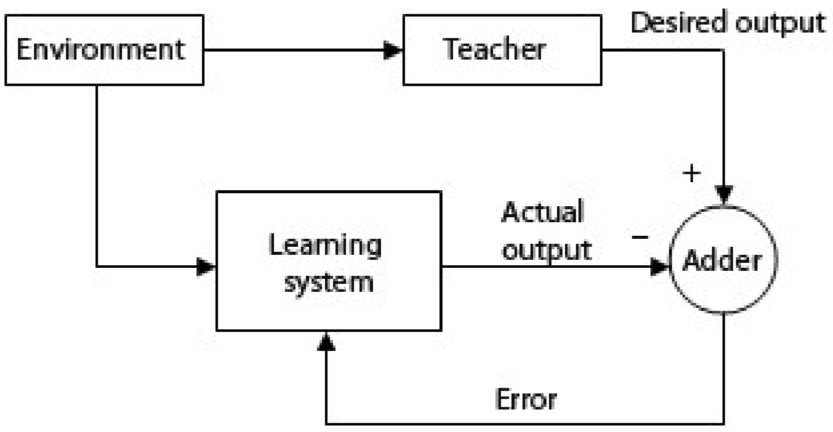


Figure 2: Back Propagation Algorithm

The back propagation learning algorithm can be divided into two phases: propagation and weight update.

Phase 1: Propagation

Each propagation involves the following steps:

1. Forward propagation of a training pattern's input through the neural network in order to generate the propagation's output activations.
2. Backward propagation of the propagation's output activations through the neural network using the training pattern target in order to generate the deltas (the difference between the targeted and actual output values) of all output and hidden neurons.

Phase 2: Weight update

For each weight-synapse follow the following steps:

1. Multiply its output delta and input activation to get the gradient of the weight.
2. Subtract a ratio (percentage) from the gradient of the weight.

This ratio (percentage) influences the speed and quality of learning; it is called the *learning rate*. The greater the ratio, the faster the neuron trains; the lower the ratio, the more accurate the training is. The sign of the gradient of a weight indicates where the error is increasing; this is why the weight must be updated in the opposite direction.

Repeat phase 1 and 2 until the performance of the network is satisfactory.

2. Artificial Neural Network

2.1 Artificial Neural Network

Artificial Neural Networks (ANNs) are computational modeling tools that have recently emerged and found extensive acceptance in many disciplines for modeling complex real-world problems. ANNs may be defined as structures comprised of densely interconnected adaptive simple processing elements (called artificial neurons or nodes) that are capable of performing massively parallel computations for data processing and knowledge representation. Although ANNs are drastic abstractions of the biological counterparts, the idea of ANNs is not to replicate the operation of the biological systems but to make the use of what is known about the functionality of the biological networks for solving complex problems. The attractiveness of ANNs comes from the remarkable information processing characteristics of the biological system such as nonlinearity, high parallelism, robustness, fault and failure tolerance, learning, ability to handle imprecise and fuzzy information, and their capability to generalize.

Artificial models possessing such characteristics are desirable because

(i) Nonlinearity allows better fit to the data

(ii) Insensitivity provides accurate prediction in the presence of uncertain data and measurement errors

(iii) High parallelism implies fast processing and hardware failure-tolerance

(iv) Learning and adaptively allow the system to update (modify) its internal structure in response to changing environment

(v) Generalization enables application of the model to unlearned data.

The main objective of ANN-based computing (neurocomputing) is to develop mathematical algorithms that will enable ANNs to learn by mimicking information processing and knowledge acquisition in the human brain. ANN-based models are empirical in nature; however they can provide practically accurate solutions for precisely or imprecisely formulated problems and for phenomena that are only understood through experimental data and field observations. In microbiology, ANNs have been utilized in a variety of applications ranging from modeling, classification, pattern recognition, and multivariate data analysis. Sample applications include

(i) interpreting pyrolysis mass spectrometry, GC, and HPLC data,

(ii) pattern recognition of DNA, RNA, protein structure, and microscopic images,

(iii) prediction of microbial growth, biomass, and shelf life of food products, and

(iv) identification of microorganisms and molecules

2.2 ANNs and biological neural networks

Because the biological neuron is the basic building block of the nervous system, its operation will be briefly discussed for understanding artificial neuron operation and the analogy between ANNs and biological networks.

2.2.1 Biological neuron

The human nervous system consists of billions of neurons of various types and lengths relevant to their location in the body. Figure 2 shows a schematic of an oversimplified biological neuron with three major functional units – dendrites, cell body, and axon. The cell body has a nucleus that contains information about hereditary traits, and plasma that holds the molecular equipment used for producing the material needed by the neuron. The dendrites receive signal from other neurons and pass them over to the cell body.

The total receiving area of the dendrites of a typical neuron is approximately 0.25 mm. The axon, which branches into collaterals, receives signals from the cell body and them away through the synapse (a micro gap) to the dendrites of neighboring neurons. A schematic illustration of the signal transfer be two neurons through the synapse is shown in Figure 3 (b). An impulse, in the form of an electric signal, travels within the dendrites and through the cell body towards the pre-synaptic membrane of the synapse. Upon arrival at the membrane, a neurotransmitter (chemical) is released from the vesicles in quantities proportional to the strength of the incoming signal. The neurotransmitter diffuses within the synaptic gap towards the post-synaptic membrane, and eventually into the dendrites of neighboring neurons, thus them (depending on the threshold of the receiving neuron) to generate a new electrical signal. The generated signal passes through the second

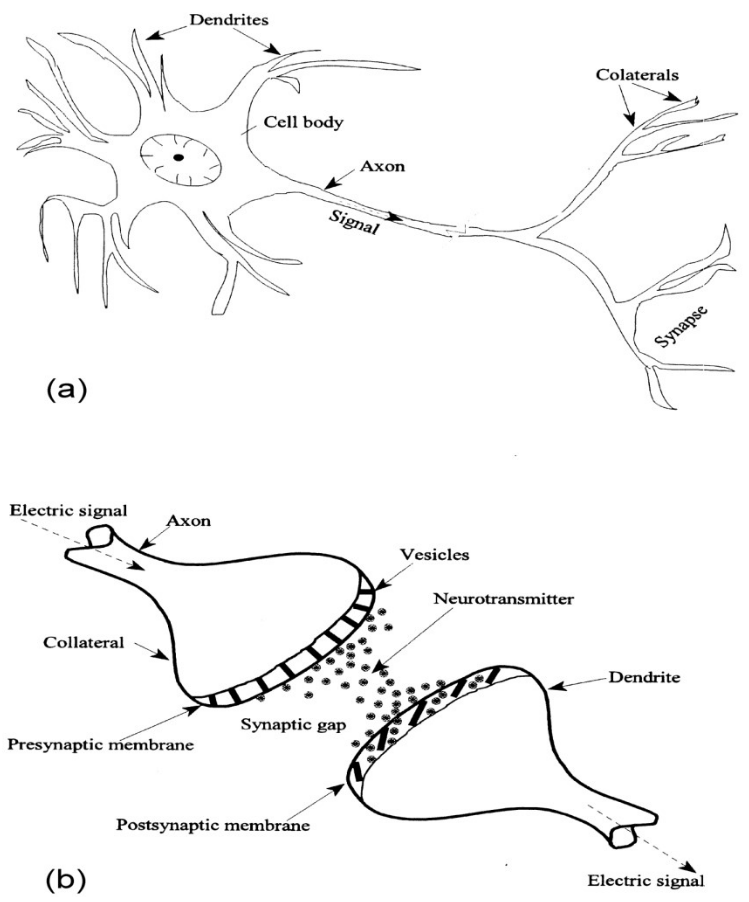


Figure 4 (a) Schematic of biological neuron. (b) Mechanism of signal transfer between two biological neurons.

neuron(s) in a manner identical to that just described. The amount of signal that passes through receiving neuron depends on the intensity of the signal emanating from each of the feeding neurons, their synaptic strengths, and the threshold of the receiving neuron. Because a neuron has a large number of dendrites / synapses, it can receive and transfer many signals simultaneously. These signals may either assist (excite) or inhibit the firing of the neuron. This simplified mechanism of signal transfer constituted the fundamental step of early neurocomputing development (e.g., the binary threshold unit of McCulloh and Pitts, 1943) and the operation of the building unit of ANNs.

2.2.2 Analogy

The crude analogy between artificial neuron and biological neuron is that the connections between nodes represent the axons and dendrites, the connection weights represent the synapses, and the threshold approximates the activity in the soma Figure 4 illustrates *n* biological neurons with various signals of intensity *x* and synaptic strength *w* feeding into a neuron with a threshold of *b*, and the equivalent artificial neurons system. Both the biological network and ANN learn by incrementally adjusting the magnitudes of the weights or synapses strengths .

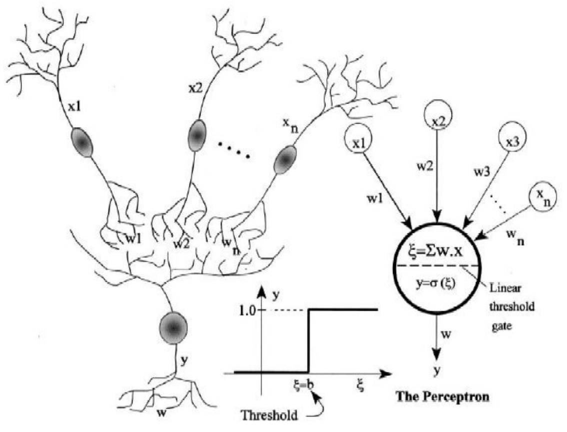


Figure 4 : Signal interaction from n neurons and analogy to signal summing in an artificial neuron comprising the single layer perceptron.

2.2.3. Artificial neuron

In 1958, Rosenblatt introduced the mechanics of the single artificial neuron and introduced the 'Perception' to solve problems in the area of character recognition . Basic findings from the biological neuron operation enabled early researchers to model the operation of simple artificial neurons. An artificial processing neuron receives inputs as stimuli from the environment, combines them in a special way to form a 'net' input(j), passes that over through a linear threshold gate, and transmits the (output, *y*) signal forward to another neuron or the environment, as shown in Figure 4. Only when j exceeds (i.e., is stronger than) the neuron's threshold limit (also called bias, *b*), will the neuron fire (i.e. becomes activated). Commonly, linear neuron dynamics are assumed for calculating j The net input is computed as the inner (dot) product of the input signals (*x*) impinging on the neuron and their strengths (*w*). For *n* signals, the neuron operation is expressed as

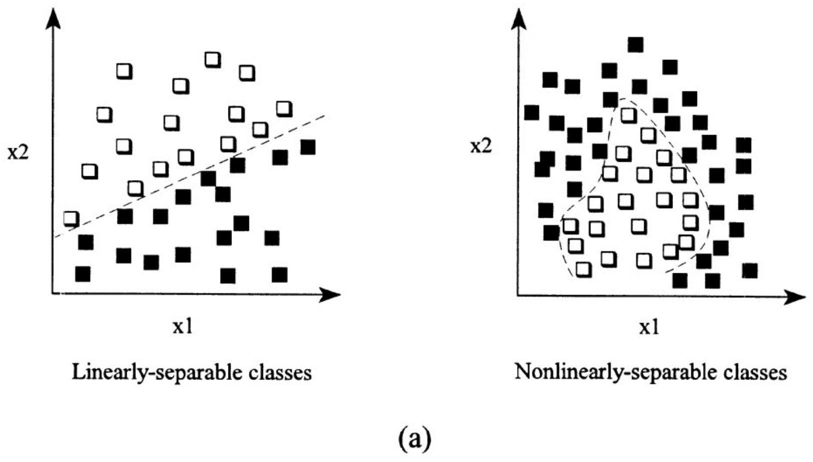
$$y = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} w_i x_i \geq b, \\ 0, & \text{if } \sum_{i=1}^{n} w_i x_i < b. \end{cases}$$

Equation 1

With 1 indicating 'on' and 0 indicating 'off' (Figure 2), or class A and B, respectively, in solving classification problems. Positive connection weights ($w_i > 0$) enhance the net signal (j) and excite the neuron, and the link is called excitory, whereas negative weights reduce j and inhibit the neuron activity, and the link is called inhibitory. The system comprised of an artificial neuron and the inputs as shown in Figure 4 is called the Perceptron which establishes a mapping between the inputs activity (stimuli) and the output In Equation. (1), the neuron threshold may be considered as an additional input node whose value always unity (i.e., x=1) and its connection weight is equal to *b*. In such case, the summation in Equation. (1) is run from 0 to n, and the net signal j is compared to 0.

2.2.4. Perceptrons

The perceptron (Figure 4) can be trained on a set of examples using a special learning rule. The perceptron weights (including the threshold) are changed in proportion to the difference (error) between the target (correct) outputs, *Y*, and the perceptron solution, *y*, for each example. The error is a function of all the weights and forms an irregular multidimensional complex hyper plane with many peaks, saddle points, and minima. Using a specialized search technique, the learning process strives to obtain the set of weights that corresponds to the global minimum. Rosenblatt (1962) derived the perceptron rule that will yield an optimal weight vector in a finite number of iterations, regardless of the initial values of the weights.



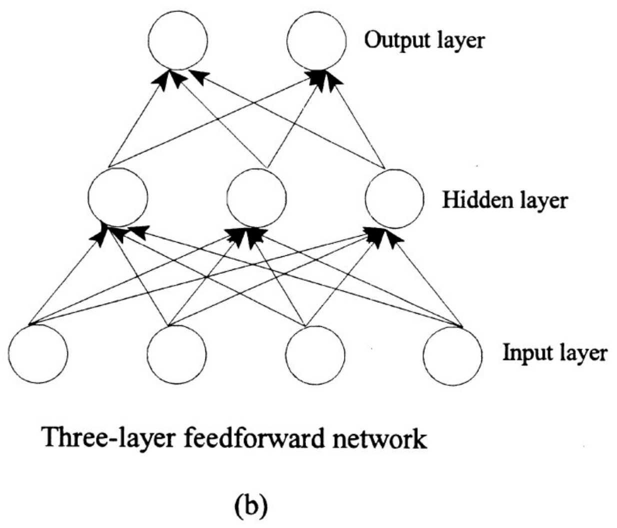


Figure 5 (a) Linear vs. nonlinear separability. (b) Multilayer perceptron showing input, hidden, and output layers and nodes with feed forward links.

This rule, however, can only perform accurately with linearly separable classes, in which a linear hyper plane can place one class of objects on one side of the plane and the other class on the other side. Fig. 5a shows linearly and nonlinearly separable two-object classification problems. In order to cope with nonlinearly separable problems, additional layer(s) of neurons placed between the input layer (containing input nodes) and the *n* output neuron are needed leading to the multilayer perceptron (MLP) architecture (Hecht-

Nielsen, 1990), as shown in Fig. 5b. Because these intermediate layers do not interact with the external environment, they are called hidden layers and their nodes called hidden nodes. The addition of intermediate layers revived the perceptron by extending its ability to solve nonlinear classification problems.

Using similar neuron dynamics, the hidden neurons process the information received from the input nodes and pass them over to output layer. The learning of MLP is not as direct as that of the simple perceptron. For example, the back propagation network is one type of MLP trained by the delta learning rule. However, the learning procedure is an extension of the simple perceptron algorithm so as to handle the weights connected to the hidden nodes.

2.2.5. Biological vs. artificial network

Central to our biological neural network is the cerebral cortex (cerebrum) which is a 2–3 mm thick flat sheet of massively interconnected neurons with an approximate surface area of 2200 cm containing about 10 11neurons (Jain et al., 1996). Each neuron is connected to 1000–10,000 other, making approximately 10 to 10 interconnections. In contrast, ANNs typically range from 10 to as high as 10,000 neurons for the most sophisticated networks implementable on a digital computer, with a connection density ranging from five to 100 links per neuron with regard to their operation and structure, ANNs are considered homogenous and often operate deterministically, whereas those of the human cortex are extremely heterogeneous and operate in a mixture of complex deterministic and stochastic manner. With regard to functionality, it is not surprising to see that ANNs compare, though roughly, to biological networks as they are developed to mimic the computational properties of the brain such as adaptively, noise (data) and fault (neurons and connections lost) tolerance.

2.2.6. Learning

The ability to learn is a peculiar feature pertaining to intelligent systems, biological or otherwise. In artificial systems, learning is viewed as the process of updating the internal representation of the system in response to external stimuli so that it can perform a specific task. This includes modifying the network architecture, which involves adjusting the weights of the links, pruning or creating some connection links, and/or changing the firing rules of the individual neurons. ANN learning is performed iteratively as the network is presented with training examples, similar to the way we learn from experience. An ANN-based system is said to have learnt if it can

(i) Handle imprecise, fuzzy, noisy, and probabilistic information without noticeable adverse effect on response quality, and

(ii) Generalize from the tasks it has learned to unknown ones.

2.3. Challenging problems

Generally, ANNs are more robust and often outperform other computational tools in solving a variety of problems from seven categories.

2.3.1. Pattern classification

Pattern classification deals with assigning an un known input pattern, using supervised learning, to one of several prespecified classes based on one or more properties that characterize a given class, as shown in Figure 6a. Classification applications from the area of microbiology include classification of commodities based on their microbiological characteristics, and characterization of microorganisms using pyrolysis mass spectrometry data. Unlike discriminate analysis in statistics, ANNs do not require the linearity assumption and can be applied to nonlinearly separable classes (Garth et al., 1996).

2.3.2. Clustering

Clustering is performed via unsupervised learning in which the clusters (classes) are formed by exploring the similarities or dissimilarities between the input patterns based on their inter-correlations (Figure 6b). The network assigns similar' patterns to the same cluster. Example applications from microbiology include sub-species discrimination using pyrolysis mass spectrometry and Kohonen networks.

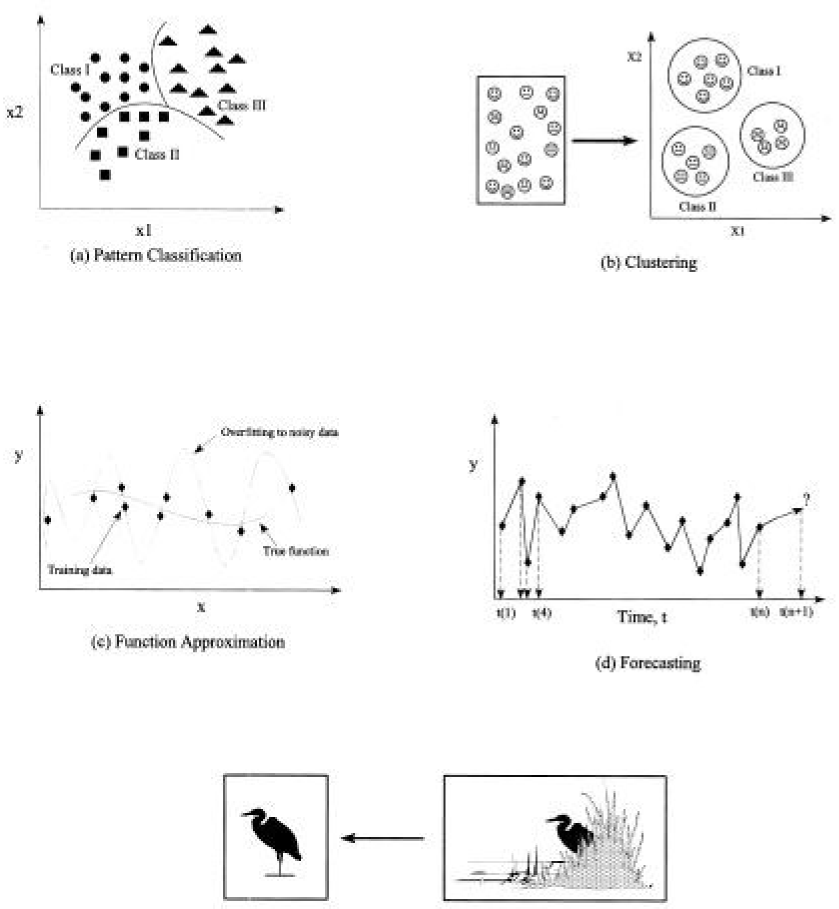


Figure 6: Problems solved by ANNs. (a) Pattern classification. (b) Clustering. (c) Function approximation. (d) Forecasting. (e) Association

2.3.3. Function approximation (modeling)

Function approximation (modeling) involves training ANN on input–output data so as to approximate the underlying rules relating the inputs to the outputs (Figure 6c). Multilayer ANNs are considered universal approximates that can approximate any arbitrary function to any degree of accuracy, and thus are normally used in this application. Function approximation is applied to problems (i) where no theoretical model is available, i.e., data obtained from experiments or observations are utilized, or (ii) to substitute theoretical models that are hard to compute analytically by utilizing data obtained from such models. Examples from this category are numerous in microbiology, e.g., predicting microbial growth.

2.3.4. Forecasting

Forecasting includes training of an ANN on samples from a time series representing a certain phenomenon at a given scenario and then using it for other scenarios to predict (forecast) the behavior at subsequent times (Figure6d). That is, the network will predict *Y*(*t* + 1) from one or more previously known historical observations [e.g., *Y*(*t*2-2), *Y*(*t*2-1), and *Y*(*t*), where *t* is the time step]. Microbial growth curves can be modeled in such a manner.

2.3.5 Optimization

Optimization is concerned with finding a solution that maximizes or minimizes an objective function subject to a set of constraints. Optimization is a well-established field in mathematics; however ANNs, such as the Hopfield network were found to be more efficient in solving complex and nonlinear optimization problems.

2.3.6 Association

Association involves developing a pattern association ANN by training on ideal noise free data and subsequently using this ANN to classify noise corrupted data (e.g., for novelty detection). The associative network may also be used to correct (reconstruct) the corrupted data or completely missing data (or image), as shown in Figure 6e. Hopfield and Hamming networks are especially used for this application (Lippmann, 1987), and to a lesser degree multilayer back propagation ANNs trained on pat- terns with identical input and output.

2.3.7 Control

Control is concerned with designing a network, normally recurrent, that will aid an adaptive control system to generate the required control inputs such system to generate the required control inputs such on system feedback.

2.4 Classification of ANNs

ANNs may be classified in many different ways according to one or more of their relevant features. Generally, classification of ANNs may be based on

(i) The function that the ANN is designed to serve (e.g., pattern association, clustering).

(ii) The degree (partial / full) of connectivity of the neurons in the network

(iii) The direction of flow of information within the network (recurrent and non recurrent), with recurrent networks being dynamic systems in which the state at any given time is dependent on previous states.

(iv) The type of learning algorithm, which represents a set of systematic equations that utilize the outputs obtained from the network along with an arbitrary performance measure to update the internal structure of the ANN.

(v) The learning rule (the driving engine of the learning algorithm).

(vi) The degree of learning supervision needed for ANN training.

Supervised learning involves training of an ANN with the correct answers (i.e., target outputs) being given for every example, and using the deviation error) of the ANN solution from corresponding target values to determine the required amount by which each weight should be adjusted. Reinforcement learning is supervised, however the ANN is provided with a critique on correctness of output rather than the correct answer itself. Unsupervised learning does not require a correct answer for the training examples, however the network, through exploring the underlying structure in the data and the correlation between the various examples, organizes the examples into clusters (categories) based on their similarity or dissimilarity (e.g., Kohonen networks). Finally, the hybrid learning procedure combines supervised and unsupervised learning. As examples of classification, Lippmann (1987) classified ANNs with regard to learning (supervised vs. unsupervised) and data (binary vs. continuous). Simpson (1990) categorized ANNs with respect to learning (supervision) and the flow of data in the network (feed forward vs. feedback). Maren (1991) proposed a hierarchal categorization based on structure followed by dynamics, then learning. Jain et al. (1996) present a four-level classification based on the degree of learning supervision, the learning rule, data flow in the ANN, and the learning algorithm.

2.5 Learning rules

A learning rule defines how exactly the network weights should be adjusted (updated) between successive training cycles (epochs). There are four basic types of rules.

2.5.1 Error-Correction Learning (ECL)

The error-correction learning (ECL) rule is used in supervised learning in which the arithmetic difference (error) between the ANN solution at any stage (cycle) during training and the corresponding correct answer is used to modify the connection weights so as to gradually reduce the overall network error.

2.5.2 The Boltzmann learning (BL)

The Boltzmann learning (BL) rule is a stochastic rule derived from thermodynamic principles and information. It is similar to ECL, however each neuron generates an output (or state) based on a Boltzmann statistical distribution, which renders learning extremely slower.

2.5.3 The Hebbian learning (HL)

The Hebbian learning (HL) rule developed based on neurobiological experiments, is the oldest learning rule, which postulates that ‘‘if neurons on both sides of a synapse are activated synchronously and repeatedly, the synapse’s strength is selectively increased.’’ Therefore, unlike ECL and BL rules, learning is done locally by adjusting the synapse weight based on the activities of the neurons.

2.5.4 Competitive learning (CL)

In the competitive learning (CL) rule, all neurons are forced to compete among themselves such that only one neuron will be activated in a given iteration with all the weights attached to it adjusted. The CL rule is speculated to exist in many biological systems.

2.6 Popular ANNs

A vast number of networks, new or modifications of existing ones, are being constantly developed. Simpson (1990) listed 26 different types of ANNs, and Maren (1991) listed 48. Pham (1994) estimated that over 50 different ANN types exist. Some applications may be solved using different ANN types, whereas others may only be solved via a specific ANN type. Some networks are more proficient in solving perceptual problems, while others are more suitable for data modeling and function approximation. A brief discussion of the most frequently used ANNs, presented in the order of their discovery, is given below.

2.6.1. Hopfield networks

This network is a symmetric fully connected two layer recurrent network that acts as a nonlinear associative memory and is especially efficient in solving optimization problems. The network is suited to only bipolar or binary inputs and it implements an energy function. Learning is done by setting each weight connecting two neurons to the product of the inputs of these two neurons. When presented with an incomplete or noisy pattern, the network responds by retrieving an internally stored pattern that most closely resembles the presented pattern.

2.6.2. Adaptive resonance theory (ART) networks

These are trained by unsupervised learning where the network adapts to the information environment without intervention. The ART network consists of two fully interconnected layers, a layer that receives the inputs and a layer consisting of output neurons. The feed forward weights are used to select the winning output neuron (cluster) and serve as the long-term memory for the networks. The feedback weights are the vigilance weights that are used to test the vigilance and serve as the short-term memory for the network. An ART network stores a set of patterns in such a way that when the network is presented with a new pattern it will either match it to a previously stored pattern, or store it as a new pattern if it is sufficiently dissimilar to the closest. Like Hopfield nets, ART networks can be used for pattern recognition, completion, and classification.

2.6.3. Kohonen networks

These networks, also called self-organizing feature maps, are two-layer networks that transform *n*-dimensional input patterns into lower-ordered data where similar patterns project onto points in close proximity to one another. Kohonen networks are trained in an unsupervised manner to form clusters within the data (i.e., data grouping). In addition to pattern recognition and classification, Kohonen maps are used for data compression, in which high-dimensional data are mapped into a fewer dimensions space while preserving their content .

2.6.4. Backpropagation networks

These networks are the most widely used type of networks and are considered the workhorse of ANNs. A back propagation (BP) network is an MLP consisting of,

(i) An input layer with nodes representing input variables to the problem

(ii) An output layer with nodes representing the dependent variables (i.e., what is being modeled)

(iii) One or more hidden layers containing nodes to help capture the nonlinearity in the data.

Using supervised learning (with the ECL rule), these networks can learn the mapping from one data space to another using examples. The term back propagation refers to the way the error computed at the output side is propagated backward from the output layer, to the hidden layer, and finally to the input layer. In BPANNs, the data are fed forward into the network without feedback (i.e., all links are unidirectional and there are no same layer neuron-to-neuron connections). The neurons in BPANNs can be fully or partially interconnected. These networks are so versatile and can be used for data modeling, classification, forecasting, control, data and image compression, and pattern recognition.

2.6.5. Recurrent networks

In a recurrent network, the outputs of some neurons are fed back to the same neurons or to neurons in preceding layers. This enables a flow of information in both forward and backward directions, thus providing the ANN with a dynamic memory There are special algorithms for training recurrent . The BP recurrent ANNs are a simple variant of recurrent networks in which the 'memory' is introduced into static feedforward ANNs by a special data representation (e.g., time delay) followed by training using classic BP.

2.6.6. Counterpropagation networks

These networks, developed by Hecht-Nielsen (1988, 1990), are trained by hybrid learning to create a self-organizing look-up table useful for function approximation and classification. As input features are presented to the network, unsupervised learning is carried out to create a Kohonen map of the input data. Meanwhile, supervised learning is used to associate an appropriate output vector with each point on the map. Once the network has been trained, each newly presented feature vector will trigger a response which is the average for those feature vectors closest to it in the input data space, thus simulating a look-up table.

2.6.7. Radial basis function (RBF) networks

These networks are a special case of a multilayer feedforward error-backpropagation network with three layers .They can be trained by a variety of learning algorithms including a two step hybrid learning .The hidden layer is used to cluster the inputs of the network (the nodes in this layer are called cluster centers). Unlike the sigmoid transfer function in BPANNs, these networks employ a radial basis function such as a Gaussian kernel. The RBF is centered at the point specified by the weight vector associated with the unit. Both the positions and widths of these Gaussian functions must be learnt from the training patterns. Each output unit implements a linear combination of these RBFs. The choice between the RBF networks and the BPANNs is problem dependent. RBF networks train faster than BP but are not as versatile and are comparatively slower for use. The decision as to which network works better for a given problem depends strictly on the problem logistics. For example, a clustering problem requires a Kohonen network, a mapping problem may be modeled using a variety of ANNs such as BP and RBF networks, and some optimization problems may only be solved using Hopfield networks. Other factors governing ANN selection are the input type (i.e., whether it is Boolean, continuous, or a mixture), and the execution speed of the network once trained and implemented in serial hardware. Other issues for ANN selection are discussed by Hudson and Postma

2.8. Back propagation ANNs

To extend the understanding of ANNs from the level of identifying what these systems are to how to design them, it is imperative to become familiar with ANN computation and design. For this objective, the BPANNs are discussed in more detail, for their popularity, and their flexibility and adaptability in modeling a wide spectrum of problems in many application areas. The feed forward error back propagation learning algorithm is the most famous procedure for training ANNs. BP is based on searching an error surface error as a function of ANN weights) using gradient descent for point(s) with minimum error. Each iteration in BP constitutes two sweeps: forward activation to produce a solution, and a backward propagation of the computed error to modify the weights. In an initialized ANN (i.e., an ANN with assumed initial weights), the forward sweep involves presenting the network with one training example. This starts at the input layer where each input node transmits the value received forward to each hidden node in the hidden layer. The collective effect on each of the hidden nodes is summed up by performing the dot product of all values of input nodes and their corresponding interconnection weights, as described in Equation. (1). Once the net effect at one hidden node is determined, the activation at that node is calculated using a transfer function (e.g., sigmoidal function) to yield an output between 0 and 11 or 21 and 11. The amount of activation obtained represents the new signal that is to be transferred forward to the subsequent layer (e.g., either hidden or output layer). The same procedure of calculating the net effect is repeated for each hidden node and for all hidden layers. The net effect(s) calculated at the output node(s) is consequently transformed into activation(s) using a transfer function. The activation(s) just calculated at the output node(s) represents the ANN solution of the fed example, which may deviate considerably from the target solution due to the arbitrary selected interconnection weights. In the backward sweep, the difference (i.e., error) between the ANN and target outputs is used to adjust the interconnection weights, starting from the output layer, through all hidden layers, to the input layer, as will be described in the following section. The forward and backward sweeps are performed repeatedly until the ANN solution agrees with the target value within a pre specified tolerance. The BP learning algorithm provides the needed weight adjustments in the backward sweep.

2.9. Back Propagation (BP) Algorithm

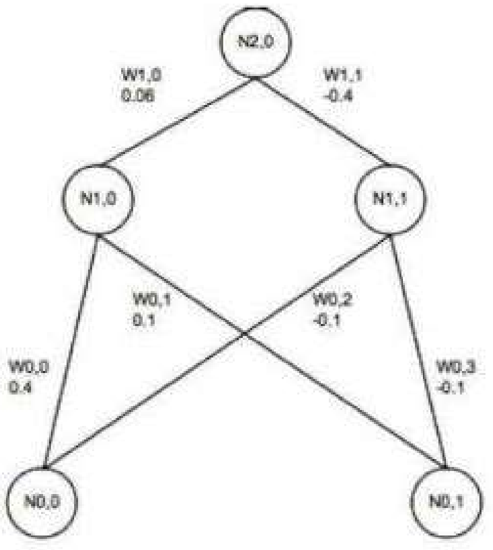
One of the most popular NN algorithms is back propagation algorithm. Rojas claimed that BP algorithm could be broken down to four main steps. After choosing the weights of the network randomly, the back propagation algorithm is used to compute the necessary corrections. The algorithm can be decomposed in the following four steps:

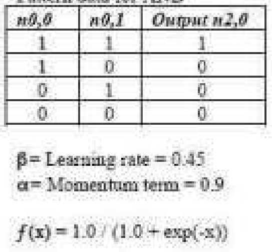
i) Feed-forward computation

ii) Back propagation to the output layer

iii) Back propagation to the hidden layer

iv) Weight updates The algorithm is stopped when the value of the error function has become sufficiently small.





Pattern data for AND

n0,0	*n0,1*	*Output n2,0*
1	1	1
1	0	0
0	1	0
0	0	0

β= Learning rate = 0.45
α= Momentum term = 0.9

$f(x) = 1.0 / (1.0 + \exp(-x))$

Figure 7 Example

This is very rough and basic formula for BP algorithm. There are some variation proposed by other scientist but Rojas definition seem to be quite accurate and easy to follow. The last step, weight updates is happening throughout the algorithm. BP algorithm will be explained using exercise example from Figure 7

2.9.1 Worked example

NN on figure 4 has two nodes (N0,0 and N0,1) in input layer, two nodes in hidden layer (N1,0 and N1,1) and one node in output layer (N2,0). Input layer nodes are connected to hidden layer nodes with weights (W0,1-W0,4). Hidden layer nodes are connected with output layer node with weights (W1,0 and W1,1). The values that were given to weights are taken randomly and will be changed during BP iterations. Table with input node values and desired output with learning rate and momentum are also given in figure 7. There is also sigmoid function formula *f(x) = 1.0/(1.0 + exp(−x))*. Shown are calculations for this simple network (only calculation for example set 1 is going to be shown (input values of 1 and 1 with output value 1)). In NN training, all example sets are calculated but logic behind calculation is the same.

2.9.1.1 Feed-forward computation

Feed forward computation or forward pass is two step process. First part is getting the values of the hidden layer nodes and second part is using those values from hidden layer to compute value or values of output layer. Input values of nodes N0,0 and N0,1 are pushed up to the network towards nodes in hidden layer (N1,0 and N1,1). They are multiplied with weights of connecting nodes and values of hidden layer nodes are calculated. Sigmoid function is used for calculations f(x) = 1.0/(1.0 + exp(−x)). N1, 0 = f(x1) = f(w0, 0 ∗ n0, 0 + w0, 1 ∗ n0, 1) = f(0.4 + 0.1) = f(0.5) = 0.622459 N1, 1 = f(x2) = f(w0, 2 ∗ n0, 0 + w0, 3 ∗ n0, 1) = f(−0.1 − 0.1) = f(−0.2) = 0.450166

When hidden layer values are calculated, network propagates forward, it propagates values from hidden layer up to a output layer node (N2,0). This is second step of feed forward computation $N2, 0 = f(x3) = f(w1, 0 * n1, 0 + w1, 1 * n1, 1) = f(0.06 * 0.622459 + (-0.4) * 0.450166) = f(-0.1427188) = 0.464381$ Having calculated N2,0, forward pass is completed.

2.9.1.2 Back propagation to the output layer

Next step is to calculate error of N2,0 node. From the table in figure 4, output should be 1. Predicted value (N2,0) in our example is 0.464381. Error calculation is done the following way: $N2, 0Error = n2, 0*(1-n2, 0)*(N2, 0Desired-N2, 0) = 0.464381(1-0.464381)*(1-0.464381) = 0.133225$ Once error is known, it will be used for backward propagation and weights adjustment. It is two step process. Error is propagated from output layer to the hidden layer first. This is where learning rate and momentum are brought to equation. So weights W1,0 and W1,1 will be updated first. Before weights can be updated, rate of change needs to be found. This is done by multiplication of the learning rate, error value and node N1,0 value. $\Delta W1, 0 = \beta * N2, 0Error * n1, 0 = 0.45 * 0.133225 * 0.622459 = 0.037317$ Now new weight for W1,0 can be calculated. $W1, 0N ew = w1, 0Old + \Delta W1, 0 + (\alpha * \Delta(t - 1)) = 0.06 + 0.037317 + 0.9 * 0 = 0.097137$ $\Delta W1, 1 = \beta * N2, 0Error * n1, 1 = 0.45 * 0.133225 * 0.450166 = 0.026988$ $W1, 1N ew = w1, 1Old + \Delta W1, 1 + (\alpha * \Delta(t - 1)) = -0.4 + 0.026988 = -0.373012$ The value of $\Delta(t - 1)$ is previous delta change of the weight. In our example, there is no previous delta change so it is always 0. If next iteration were to be calculated, this would have some value.

2.9.1.3 Back propagation to the hidden layer

Now errors has to be propagated from hidden layer down to the input layer. This is bit more complicated than propagating error from output to hidden layer. In previous case, output from node N2,0 was known beforehand. Output of nodes N1,0 and N1,1 was unknown. Let's start with finding N1,0 error first. This will be calculated multiplying new weight W1,0 value with error for the node N2,0 value. Same way error for N1,1 node will be found.

$N1, 0Error = N2, 0Error * W1, 0New = 0.133225 * 0.097317 = 0.012965$

$N1, 1Error = N2, 0Error * W1, 1New = 0.133225 * (-0.373012) = -0.049706$

Once error for hidden layer nodes is known, weights between input and hidden layer can be updated. Rate of change first needs to be calculated for every weight:

$\Delta W0, 0 = \beta * N1, 0Error * N0.0 = 0.45 * 0.012965 = 0.005834$

$\Delta W0, 1 = \beta * N1, 0Error * n0, 1 = 0.45 * 0.012965 * 1 = 0.005834$

$\Delta W0, 2 = \beta * N1, 1Error * n0, 0 = 0.45 * -0.049706 * 1 = -0.022368$

$\Delta W0, 3 = \beta * N1, 1Error * n0, 1 = 0.45 * -0.049706 * 1 = -0.022368$

Than we calculate new weights between input and hidden layer.

$W0, 0New = W0, 0Old + \Delta W0, 0 + (\alpha * \Delta(t - 1)) = 0.4 + 0.005834 + 0.9 * 0 = 0.405834$

$W0, 1New = w0, 1Old + \Delta W0, 1 + (\alpha * \Delta(t - 1)) = 0.1 + 0.005834 + 0 = 0.105384$

$W0, 2New = w0, 2Old + \Delta W0, 2 + (\alpha * \Delta(t - 1)) = -0.1 + -0.022368 + 0 = -0.122368$

$W0, 3New = w0, 3Old + *\Delta W0, 3 + (\alpha * \Delta(t - 1)) = -0.1 + -0.022368 + 0 = -0.122368$

3.1.4 Weight updates

Important thing is not to update any weights until all errors have been calculated. It is easy to forget this and if new weights were used while calculating errors, results would not be valid. Here is quick second pass using new weights to see if error has decreased.

$N1, 0 = f(x1)$

$= f(w0, 0 * n0, 0 + w0, 1 * n0, 1) = f(0.406 + 0.1) = f(0.506) = 0.623868314$

$N1, 1 = f(x2)$

$= f(w0, 2 * n0, 0 + w0, 3 * n0, 1) = f(-0.122 - 0.122) = f(-0.244) = 0.43930085$

$N2, 0 = f(x3)$

$= f(w1, 0 * n1, 0 + w1, 1 * n1, 1) = f(0.097 * 0.623868314 + (-0.373) * 0.43930085) = f(-0.103343991) = 0.474186972$

Having calculated N2,0, forward pass is completed.

Next step is to calculate error of N2,0 node. From the table in figure 4, output should be 1. Predicted value (N2,0) in our example is 0.464381. Error calculation is done in following way.

$$\begin{aligned} N2,0_{Error} &= n2,0*(1-n2,0)*(N2,0Desired-N2,0) \\ &= 0.474186972*(1-0.474186972)*(1-0.474186972) \\ &= 0.131102901 \end{aligned}$$

So after initial iteration, calculated error was 0.133225 and new calculated error is 0.131102. Our algorithm has improved, not by much but this should give good idea on how BP algorithm works. Although this was very simple example, it should help to understand basic operation of BP algorithm. It can be said that algorithm learned through iterations. Number of iterations in typical NN would be any number from ten to ten thousands. This is only one example set pass that could be repeated many times until error is small enough.

2.9.2 Advantages and Disadvantages

Negnevitsky argued that turning point in quest for intelligent machines was when Kasparov, world chess champion was defeated by computer in New York in May 1997. Artificial intelligence and NN have been used more and more in recent decades. Potentials in this area are huge. Here are some NN advantages, disadvantages and industries where they are being used. NN are used in cases where rules or criteria for searching an answer are not clear (that is why NN are often called black box, they can solve the problem but at times it is hard to explain how problem was solved). They found its way into broad spectrum of industries, from medicine to marketing and military just to name few. Financial sector has been known for using NN in classifying credit rating and market forecasts. Marketing is another field where NN has been used for customer classification (groups that will buy some product, identifying new markets for certain products, relationships between customer and company).

Many companies use direct marketing (sending its offer by mails) to attract customers. If NN could be employed to up the percentage of the response to direct marketing, it could save companies lot's of their revenue. At the end of the day, it's all about the money. Post offices are known to use NN for sorting the post (based on postal code recognition). Those were just few examples of where NN are being used. NN advantages are that they can adapt to new scenarios, they are fault tolerant and can deal with noisy data. Time to train NN is probably identified as biggest disadvantage. They also require very large sample sets to train model efficiently. It is hard to explain results and what is going on inside NN.

3. Super Resolution

3.1 Introduction

High-resolution images or videos are required in most digital imaging applications. Higher resolution offers an improvement of the graphic information for human perception. It is also useful for the later image processing, computer vision etc. Image resolution is closely related to the details included in any image. In general, the higher the resolution is, the more image details are presented.

3.2 Image Resolution

The resolution of a digital image can be classified in many different ways. It may refer to spatial, pixel, temporal, spectral or radiometric resolution. In the following work, it is dealt mainly with spatial resolution.

A digital image is made up of small picture elements called pixels. Spatial resolution is given by pixel density in the image and it is measured in pixels per unit area. Therefore, spatial resolution depends on the number of resolvable pixels per unit length. The clarity of the image is directly affected by its spatial resolution. The precise method for measuring the resolution of a digital camera is defined by The International Organization for Standardization (ISO) . In this method, the ISO resolution chart is sensed and then the resolution is measured as the highest frequency of black and white lines where it is still possible to distinguish the individual black and white lines. Final value is commonly expressed in lines per inch (lpi) or pixels per inch (ppi) or also in line widths per picture height (LW/PH). The standard also defines how to measure the frequency response of a digital imaging system (SFR) which is the digital equivalent of the modulation transfer function (MTF) used for analog devices.

The effort to attain the very high resolution coincides with technical limitations. Charged coupled device (CCD) or complementary metal-oxide-semiconductor (CMOS) sensors are widely used to capture two-dimensional image signals. Spatial resolution of the image is determined mainly by the number of sensor elements per unit area. Therefore, straightforward solution to increase spatial resolution is to increase the sensor density by reducing the size of each sensor element (pixel size). However, as the pixel size decreases, the amount of light impact on each sensor element also decreases and more shot noise is generated. In the literature, the limitation of the pixel size reduction without obtaining the shot noise is presented.

Another way to enhance the spatial resolution could be an enlargement of the chip size. This way seems unsuitable, because it leads to an increase in capacitance and a slower charge transfer rate . The image details (high frequency content) are also limited by the optics (lens blurs, aberration effects, aperture diffractions etc.). High quality optics and image sensors are very expensive. Super-resolution overcomes these limitations of optics and sensors by developing digital image processing techniques. The hardware cost is traded off with computational cost.

3.3 Super resolution

In most digital imaging applications, high resolution images or videos are usually desired for later image processing and analysis. The desire for high image resolution stems from two principal application areas: improvement of pictorial information for human interpretation; and helping representation for automatic machine perception. Image resolution describes the details contained in an image, the higher the resolution, the more image details. The resolution of a digital image can be classified in many different ways: pixel resolution, spatial resolution, spectral resolution, temporal resolution, and radiometric resolution. In this context, we are mainly interested in spatial resolution.

Spatial resolution: a digital image is made up of small picture elements called pixels. Spatial resolution refers to the pixel density in an image and measures in pixels per unit area. Figure 8 shows a classic test target to determine the spatial resolution of an imaging system.

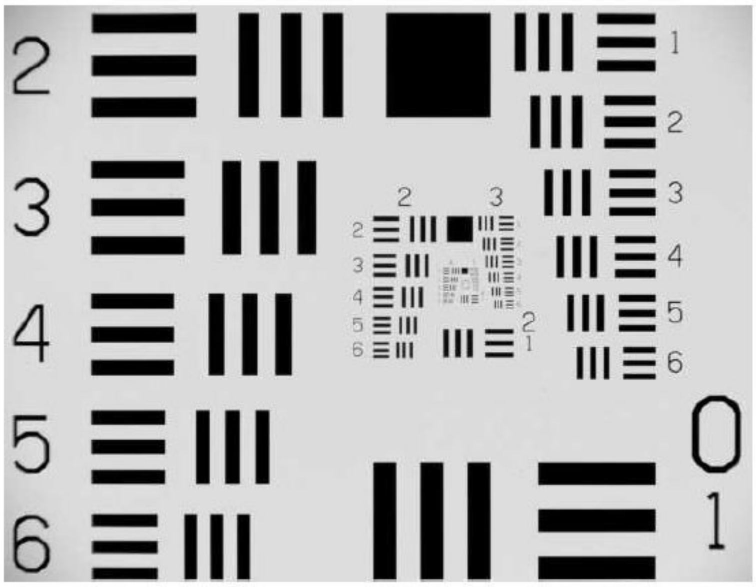


Figure 8

The 1951 USAF resolution test target, a classic test target used to determine spatial resolution of imaging sensors and imaging systems.

The image spatial resolution is firstly limited by the imaging sensors or the imaging acquisition device. Modern image sensor is typically a charge-coupled device (CCD) or a complementary metal-oxide-semiconductor (CMOS) active-pixel sensor. These sensors are typically arranged in a two-dimensional array to capture two-dimensional image signals. The sensor size or equivalently the number of sensor elements per unit area in the First place determines the spatial resolution of the image to capture. The higher density of the sensors, the higher spatial resolution possible of the imaging system. An imaging system with inadequate detectors will generate low resolution images with blocky effects, due to the aliasing from low spatial sampling frequency. In order to increase the spatial resolution of an imaging system, one straight forward way is to increase the sensor density by reducing the sensor size. However, as the sensor size decreases, the amount of light incident on each sensor also decreases, causing the so called shot noise. Also, the hardware cost of sensor increases with the increase of sensor density or correspondingly image pixel density. Therefore, the hardware limitation on the size of the sensor restricts the spatial resolution of an image that can be captured. While the image sensors limit the spatial resolution of the image, the image details (high frequency bands) are also limited by the optics, due to lens blurs (associated with the sensor point spread function (PSF)), lens aberration effects, aperture diffractions and optical blurring due to motion. Constructing imaging chips and optical components to capture very high-resolution images is prohibitively expensive and not practical in most real applications, e.g., widely used surveillance cameras and cell phone built-in cameras. Besides the cost, the resolution of a surveillance camera is also limited in the camera speed and hardware storage. In some other scenarios such as satellite imagery, it is difficult to use high resolution sensors due to physical constraints. Another way to address this problem is to accept the image degradations and use signal processing to post process the captured images, to trade of computational cost with the hardware cost. These techniques are specifically referred as Super- Resolution (SR) reconstruction.

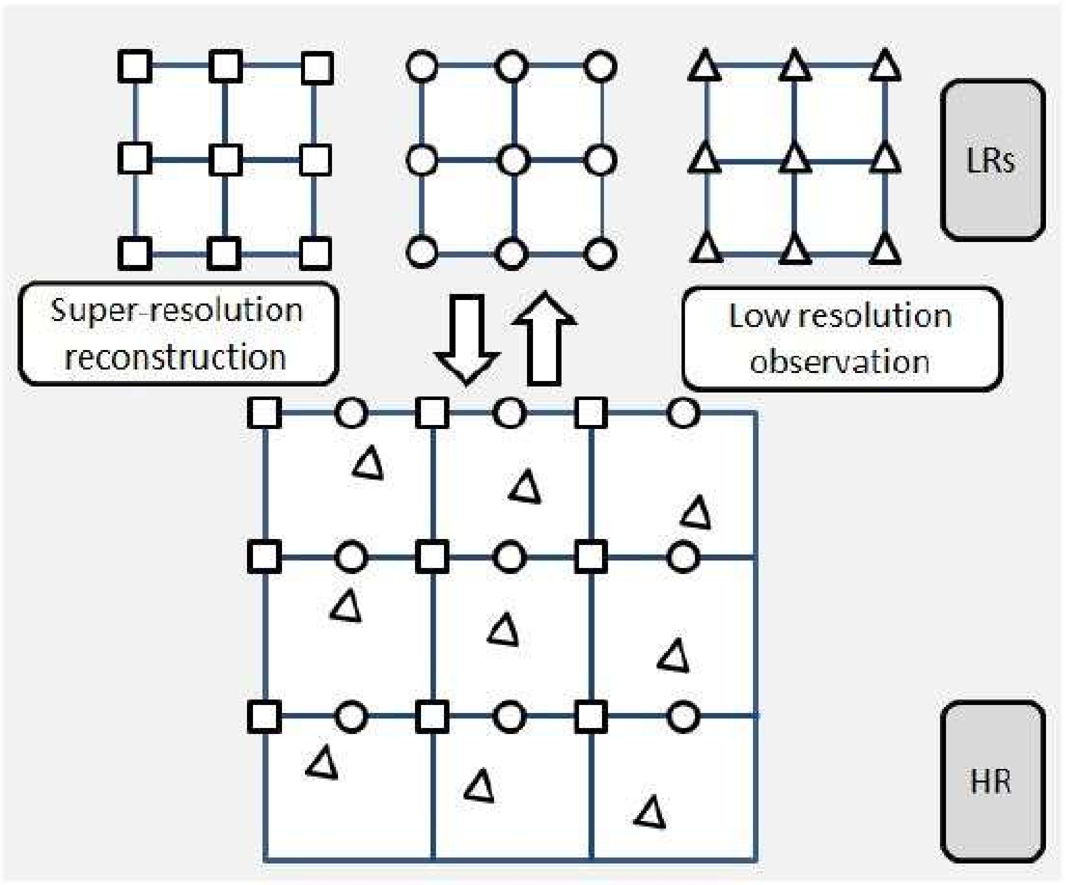
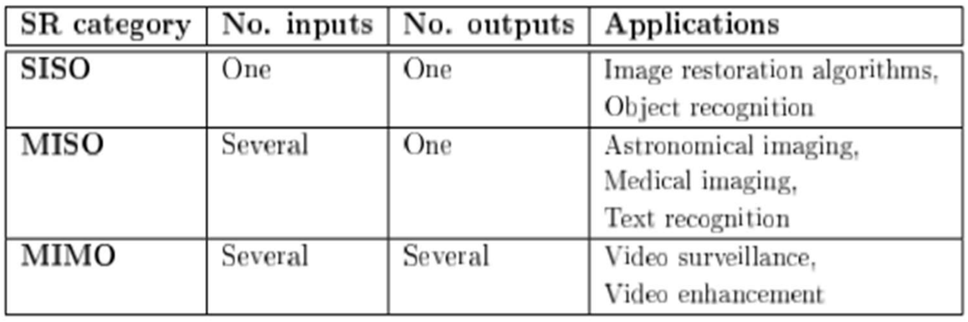


Figure 9 The basic idea for super-resolution reconstruction from multiple low-resolution

Super-resolution (SR) are techniques that construct high-resolution (HR)images from several observed low-resolution (LR) images, thereby increasing the high frequency components and removing the degradations caused by the imaging process of the low resolution camera. The basic idea behind SR is to combine the non-redundant information contained in multiple low-resolution frames to generate a high-resolution image. A closely related technique with SR is the single image interpolation approach, which can be also used to increase the image size. However, since there is no additional information provided, the quality of the single image interpolation is very much limited due to the ill-posed nature of the problem, and the lost frequency components cannot be recovered. In the SR setting, however, multiple low-resolution observations are available for reconstruction, making the problem better constrained. The non-redundant information contained in the these LR images is typically introduced by sub pixel shifts between them. These sub pixel shifts may occur due to uncontrolled motions between the imaging system and scene, e.g., movements of objects, or due to controlled motions, e.g., the satellite imaging system orbits the earth with predefined speed and path.

Each low-resolution frame is a decimated, aliased observation of the true scene. SR is possible only if there exists subpixel motions between these low resolution frames, and thus the ill-posed up sampling problem can be better conditioned. Figure 9 shows a simplified diagram describing the basic idea of SR reconstruction. In the imaging process, the camera captures several LR frames, which are down sampled from the HR scene with subpixel shifts between each other. SR construction reverses this process by aligning the LR observations to subpixel accuracy and combining them into a HR image grid (interpolation), thereby overcoming the imaging limitation of the camera.

SR algorithms can be categorized according to the number of input images and output images involved in this process. When a single high - resolution (HR) image is produced from a single degraded low resolution image we refer to single image single output super resolution(SISO). Possible applications of SISO super resolution relate to the possibility of achieving resolution enhancements, e.g. to improve relate object recognition performance and enable zoom in capabilities. Other SR algorithms deal with the integration of multiple LR frames to estimate a unique HR image: In this case we can speak about multiple input single output (MISO) super resolution. An example application area is in license plate recognition from a video stream to increase the alpha numeric recognition rate. A recent focus on SR research relates to algorithms which aim at reconstructing a set of HR frame from an equivalent set of LR frames. This approach takes the name of multiple-input single-output (MIMO) super-resolution, also known as video-to-video SR. A typical application of these algorithms can be for example the quality enhancement of a video sequence captured by surveillance cameras. Table 1 sums up the characteristics of the three SR categories described (SISO, MISO, and MISO), by mentioning for each of them possible application domains.



SR category	No. inputs	No. outputs	Applications
SISO	One	One	Image restoration algorithms, Object recognition
MISO	Several	One	Astronomical imaging, Medical imaging, Text recognition
MIMO	Several	Several	Video surveillance, Video enhancement

Table 1: Categories of SR algorithms

The first two categories of SR algorithms (SISO and MISO) give rise to two different families of SR methods, each one recalling different kinds of procedures: particularly, for SISO super-resolution we speak about single-image SR methods (a single LR image is used as input to estimate an underlying HR image), whereas, in the case of MISO super-resolution, we speak about multi-frame SR methods (the information contained within multiple under-sampled LR frames is fused to generate a single HR image). Several algorithms have been developed in the recent years for the two families of methods mentioned. MIMO super-resolution, instead, represents an emerging application, where procedures are not consolidated yet and often follow a sort of mixed approach, by borrowing elements either from SISO and MISO super-resolution algorithms. Thus, in terms of actual methodologies, we can distinguish two main families: single-image SR and multi-frame SR.

Figure 1.1 presents a taxonomic diagram of SR according to the classifications made and the relations between the different categories of SR algorithms.

SR arises in many areas such as:

1. Surveillance video : frame freeze and zoom region of interest(ROI) in video for human perception (e.g. look at the license plate in the video), resolution enhancement for automatic target recognition (e.g.try to recognize a criminal's face).

2. Remote sensing : several images of the same area are provided, and an improved resolution image can be sought.

3. Medical imaging (CT, MRI, Ultrasound etc): several images limited in resolution quality can be acquired, and SR technique canbe applied to enhance the resolution.

4. Video standard conversion, e.g. from NTSC video signal to HDTV signal.

3.4 Image observation model

The digital imaging system is not perfect due to hardware limitations, acquiring images with various kinds of degradations. For example, the finite aperture size causes optical blur, modeled by Point Spread Function (PSF). The finite aperture time results in motion blur, which is very common in videos. The finite sensor size leads to sensor blur; the image pixel is generated by integration over the sensor area instead of impulse sampling. The limited sensor density leads to aliasing effects, limiting the spatial resolution of the achieved image. These degradations are modeled fully or partially in different SR techniques.

Figure 10 shows a typical observation model relating the HR image with LR video frames, as introduced in the literature. The input of the imaging system is continuous natural scenes, well approximated as band-limited signals. These signals may be contaminated by atmospheric turbulence before reaching the imaging system. Sampling the continues signal beyond the Nyquist rate generates the high resolution digital image (a) we desire. In our SR setting, usually there exists some kind of motion between the camera and scene to capture. The inputs to the camera are multiple frames of the scene, connected by possibly local or global shifts, leading to image (b). Going through the camera, this motion related high resolution frames will incur different kinds of blurring effects, such as optical blur and motion blur. These blurred images (c) are then down sampled at the image sensors (e.g. CCD detectors) into pixels, by an integral of the image falling into each sensor area. These down sampled images are further affected by the sensor noise and color filtering noise. Finally the frames captured by the low resolution imaging system are blurred, decimated, and noisy versions of the underlying true scene.

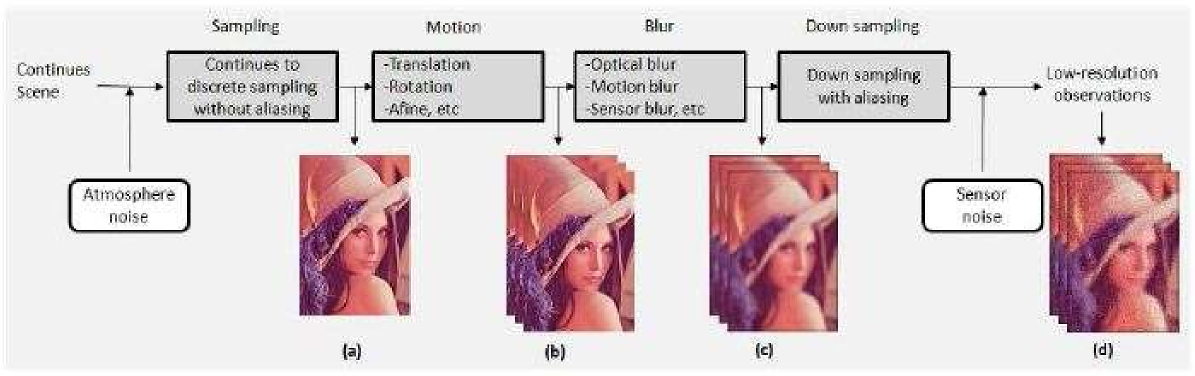


Figure 10 The observation model of a real imaging system

Let X denote the HR image desired, i.e., the digital image sampled above Nyquist sampling rate from the band limited continuous scene, and Y_k be the kth LR observation from the camera. X and $Y'_k s$ are represented in lexicographical order. Assume the camera captures K LR frames of X, where the LR observations are related with the HR scene X by

$Y_k = D_k H_k F_k X + V_k; k = 1; 2; :::;K;$

where F_k encodes the motion information for the k-th frame, H_k models the blurring effects, D_k is the down-sampling operator, and V_k is the noise term. These linear equations can be rearranged into a large linear system.

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \cdot \\ \cdot \\ Y_K \end{bmatrix} = \begin{bmatrix} D_1 H_1 F_1 \\ D_2 H_2 F_2 \\ \cdot \\ \cdot \\ D_K H_K F_K \end{bmatrix} \boldsymbol{X} + \underline{V}$$

The involved matrices D_k, H_k, F_k or M are very sparse, and this linear system is typically ill-posed. Furthermore, in real imaging systems, these matrices are unknown and need to be estimated from the available LR observations, leaving the problem even more ill-conditioned. Thus proper prior regularization for the high resolution image is always desirable and often even crucial.

3.5 Multi-frame SR methods

The basic premise for increasing the spatial resolution in multi-frame SR techniques is the availability of multiple LR images captured from the same scene. Multi-frame SR methods work effectively when several LR images contain slightly different perspectives of the scene to be super-resolved, i.e. when they represent different "looks" at the same scene. In this case, each image is seen as a degraded version of an underlying HR image to be estimated, where the degradation processes can include blurring, geometrical transformations, and down-sampling. If the geometrical transformations consist in simple shifts by integer units, then each image presents the same content and there is no extra information that can be exploited to reconstruct a common HR image.

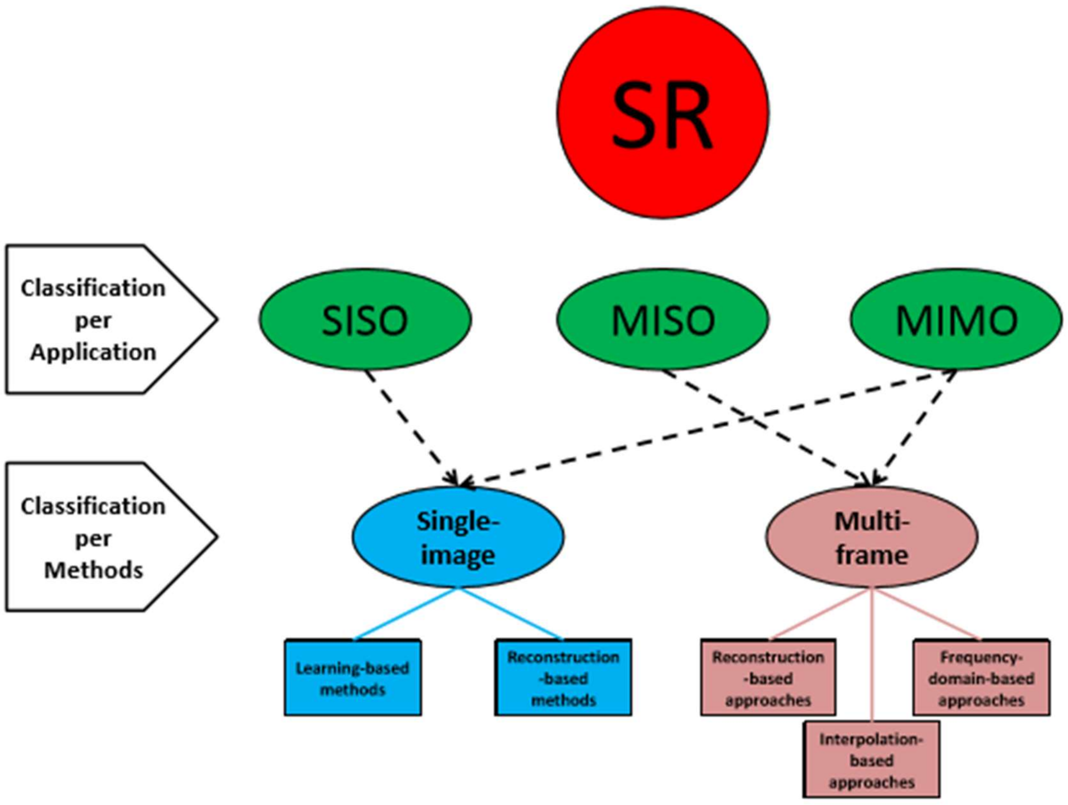

SR
Classification per Application
SISO
MISO
MIMO
Classification per Methods
Single-image
Multi-frame
Learning-based methods
Reconstruction -based methods
Reconstruction -based approaches
Frequency-domain-based approaches
Interpolation-based approaches

Figure 11: Taxonomic diagram of super-resolution

The best case for these methods to work is then when the LR images have different subpixel shifts from each other and thus they actually bring different information (each image cannot be obtained from the others).

Broadly speaking, multi-frame SR algorithms can be classified according to three main approaches followed:

1. Interpolation-based approach

2. Frequency-domain-based approach

3. Regularization-based approach

3.5.1 Interpolation-based approach

The algorithm is composed of three main stages as Figure 11 shows. Firstly, relative motion estimation between observed LR images is performed. This part is often called registration and it is crucial for success of the whole method. The estimation of relative shifts must have sub pixel precision. It has been proved that 0.2 px precision of estimation is acceptable . Pixels from registered LR images are aligned in an HR grid. After this process, points in the HR grid are non-uniformly spaced and therefore non-uniform interpolation is applied to produce an image with the enhanced resolution (HR image). When the HR image is obtained, the restoration follows to remove blurring and noise. This approach is simple and computationally efficient.

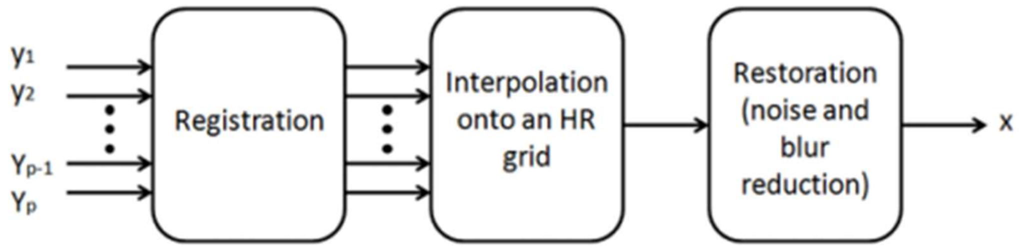

y1
y2
Yp-1
Yp
Registration
Interpolation onto an HR grid
Restoration (noise and blur reduction)
x

Figure 12 : Non-uniform interpolation approach

When LR images are aligned, a non-uniform interpolation is necessary to create a regularly sampled HR image. The basic, very simple method is the nearest neighbor interpolation. For each point in the HR grid, algorithm searches for a nearest pixel among all pixels which were aligned in the HR grid from LR images (as Figure 1 shows). The nearest pixel value is then used for the current HR grid point.

Another often used and simple method is the bilinear interpolation. At first the nearest pixel is found as in the previous case. The algorithm detects which LR image this pixel comes from and then picks up three other neighboring pixels from the same LR image. The situation is described in Figure 12. The HR grid point is surrounded by four LR pixels. These four pixels form a square so that the unknown value of the HR grid point can be calculated using the bilinear weighted sum. Similarly, the bicubic interpolation can be applied if 16 pixels in the LR image are selected (instead of four). These two methods are efficiently fast, but there is a disadvantage. Some of the 4 pixels (or 16 pixels) used for the interpolation are not among the 4 (or 16) absolutely closest pixels from all LR images. The situation is demonstrated in Figure 4. Some other pixels from other LR image are in fact closer to the HR grid point. Therefore, they may contain more relevant information about the unknown HR grid point value. Other methods are based on a selection of four closest pixels from all pixels from all input LR images (not only from a single LR image). A question remains in the determination of the weights for each of these pixels. The weights can be simply determined by a function of distance between the LR image pixel and the HR grid point.

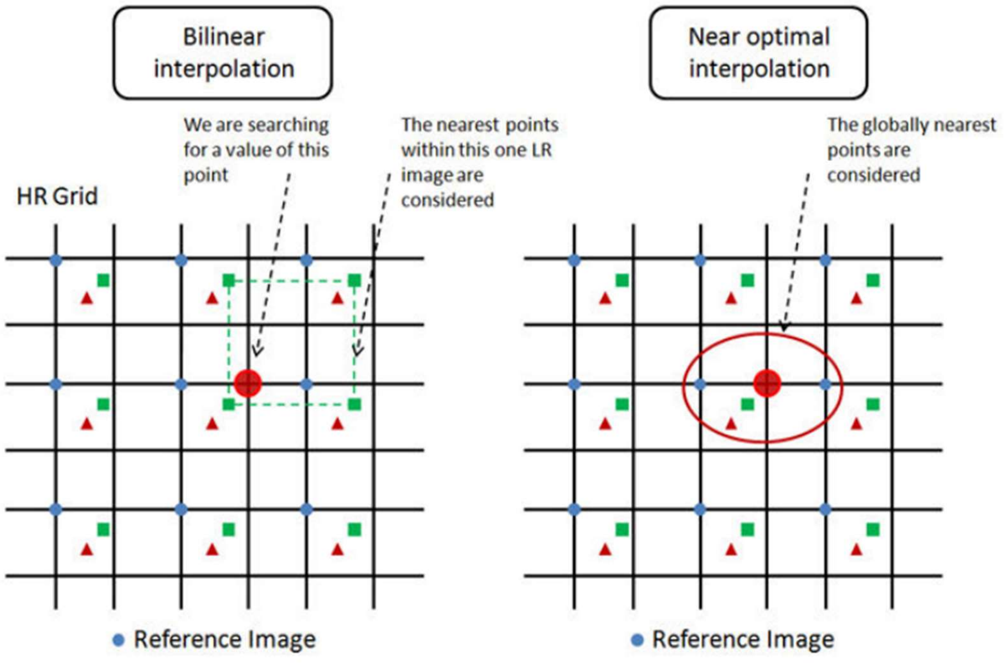

Bilinear interpolation
We are searching for a value of this point
The nearest points within this one LR image are considered
HR Grid
Reference Image
Near optimal interpolation
The globally nearest points are considered
Reference Image

Figure 13 : Bilinear non-uniform interpolation and near optimal non-uniform interpolation

Gilman and Bailey introduced near optimal non-uniform interpolation . They assume that the optimal weights depend only weakly on the image content and mostly on the relative positions of the available samples.

Therefore, the weights derived from a synthetic image can be applied to the input LR images with the same offsets. In other words, an arbitrary HR image is used to generate synthetic LR images with the same properties (size, shifts, blur) as the input LR images. The values of the weights are then derived to minimize the mean squared error between the auxiliary HR image and its version restored from the synthetic LR images. The near optimal interpolation method provides good results, but the computational cost rises rapidly if the motion model between the LR images is more complex than global translation. As a result the near optimal interpolation method as well as the bilinear interpolation method is suitable only in case of a global, pure translational movement. Lertrattanapanich and Bose used Delaunay triangulation and then fit a plane to each triangle to interpolate an HR grid point inside the triangle . The last part contains deblurring and noise removal. Restoration can be performed by applying any deconvolution method that considers the presence of noise. Wiener filtering is widely used. There is a huge amount of works dedicated to image enhancement, but that is not directly connected to SR techniques.

3.5.2 Frequency-domain-based approach

A major class of multi-frames SR methods utilizes a frequency domain formulation of the SR problem. The main principle is that clues about high frequencies are spread across the multiple LR images in form of aliased spectral frequencies. The first frequency-domain SR method can be credited to Tsai and Huang, who considered SR reconstruction from noise-free LR images. They proposed to first transform the LR image data into the Discrete Fourier Transform (DFT) domain, and then combine them according to the relationship between the aliased DFT coefficients of the observed LR images. The approach is based on the following principles:

1. The shifting property of the Fourier transform

2. The aliasing relationship between the continuous Fourier transform (CFT) and the DFT of observed LR images, and

3. The assumption that an original HR image is band-limited.

These properties make it possible to formulate an equation relating the aliased DFT coefficients of the observed LR images to a sample of the CFT of an unknown HR image.

Rhee and Kang exploited the Discrete Cosine Transform (DCT), instead of DFT, in order to reduce memory requirements and the computational costs. Woods instead, presented an iterative expectation maximization (EM) algorithm for simultaneously performing the registration, blind de convolution, and interpolation operations.

The frequency-domain-based SR approach has a number of advantages. The first advantage is its theoretical simplicity: the relationship between the LR input images and the HR image is clearly demonstrated. Thus, frequency-domain-based SR approaches represent an intuitive way to enhance the details of the images by extrapolating the high-frequency information presented in the LR images. Secondly, these approaches have low computational complexity, by also being suitable for parallel implementations. However, frequency-domain-based SR methods are insufficient to handle real-world applications, since they require that there only exists a global displacement between the observed images and the linear space invariant blur during the image acquisition process.

3.5.3 Regularization-based approach

Motivated by the fact that the SR computation is, in essence, an ill-posed inverse problem, due to the insufficient number of LR images or ill-conditioned blur operators, numerous regularization-based SR algorithms have been developed to stabilizethe inversion operation, by decreasing the number of possible solutions. The basic idea of these regularization-based SR approaches is to use the regularization strategy to incorporate some prior knowledge of the unknown HR image. The related methods can be broadly classified into two categories:

• Deterministic regularization approaches, and

• Stochastic regularization approaches.

3.7 Single-image SR methods

Single-image SR is the problem of estimating an underlying HR image, given only one observed LR image. In this case, it is assumed that there is no access to the imaging step so that the starting point is a given LR obtained according to some known or unknown conventional imaging process.

The generation process of the LR image from the original HR image that is usually considered can be written as

$$I_L = (I_H * b) \downarrow_s$$

where IL and IH are respectively the LR and HR image, b is a blur kernel the original image is convoluted with, which is typically modeled as a Gaussian blur , and the expression '↓s' denotes a down sampling operation by a scale factor of 's'. The LR image in then a blurred and down-sampled version of the original HR image.

Single-image SR aims at constructing the HR output image from as little as a single LR input image. The problem stated is an inherently ill-posed problem, as there can be several HR images generating the same LR image. Single-image SR is deeply connected with traditional "analytical" interpolation, as they share the same goal. Traditional interpolation methods (e.g. linear, bicubic, and cubic spline interpolation), by computing the missing pixels in the HR grid as averages of known pixels, implicitly impose a "smoothness" prior. However, natural images often present strong discontinuities, such as edges and corners, and thus the smoothness prior results in producing ringing and blurring artifacts in the output image. The goal of SR is thus to achieve better results, by using more sophisticated statistical priors.

Single-image SR algorithms can be broadly classified into two main categories: 1. learning-based methods, which make use of machine learning techniques and often employ a dictionary generated from an image database. 2. reconstruction-based methods, which do not use a training set but rather define constraints for the target high-resolution image to improve the quality of the reconstruction.

3.8 Example-Based Super-Resolution

Example-based single-image SR aims at reversing the image generation model , by means of a dictionary of image examples that maps locally the relation between an HR image and its LR counterpart, the latter obtained with the model . For general up scaling purposes, the examples used are typically in the form of patches, i.e. squared blocks of pixels (e.g. 3×3 or 5×5 blocks). The dictionary is then a collection of patches, which, two by two, form pairs. A pair specifically consists of a LR patch and its HR version with enriched high frequency details.

Example-based SR algorithms comprise two phases:

1. A training phase, where the above-mentioned dictionary of patches is built;

2. The proper super-resolution phase, that uses the dictionary created to upscale the input image.

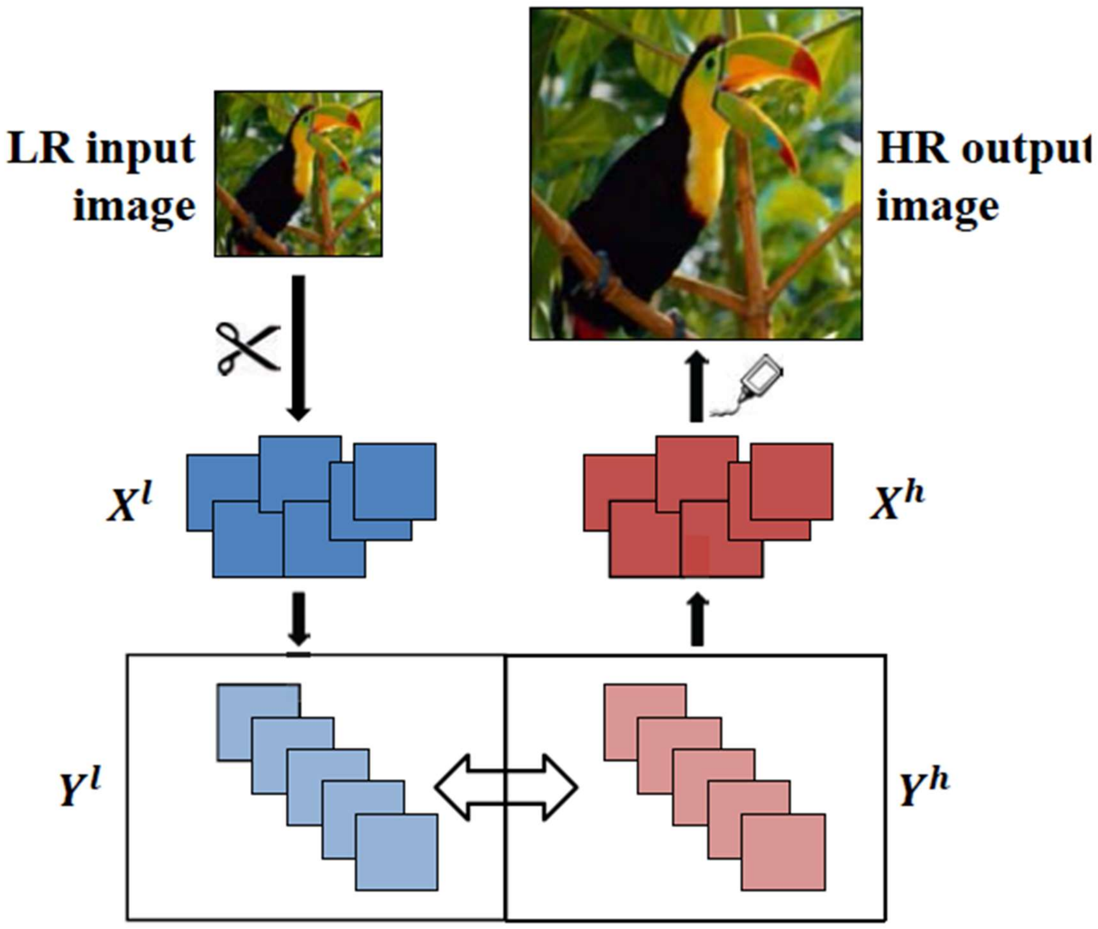


Figure 14: General Scheme of the example based SR procedure

The dictionary can be of two kinds: an external dictionary, built from a set of external training images, and an internal one, built without using any other image than the LR input image itself. This latter case exploits the so called self-similarity property, typical of natural images, according to which image structures tend to repeat within and across different image scales: therefore, patch correspondences can be found in the input image itself and possibly scaled versions of it. To learn these patch correspondences that specially take the name of self-examples, we can have one-step schemes or schemes based on a pyramid of recursively scaled images starting from the LR input image.

As for the super resolution phase, in example based algorithms the patch is also the reconstruction unit used in the up scaling procedure. In fact the LR using the LR–HR patch corresponding in the dictionary, a HR output patch is constructed, The HR output image is finally built by reassembling all the reconstructed HR patches. Figure 14 shows in a simple manner the operating diagram of an example based algorithm, according to the principle described above.

4. Implementation

4.1 Introduction

This chapter describes the implementation for developing super resolution software. Super resolution is the process of obtaining HR image from one or more LR images. This chapter includes the explanation of the environment we used for developing this software, neural network, training algorithm, results and a comparison with the existing technologies.

4.2 Software section

4.2.1 Overview of the Matlab environment

MATLAB is a high level technical computing language and interactive environment for algorithm development, data visualization, data analysis and numeric computation. Using the MATLAB product, you can solve technical computing problems faster than with traditional programming languages, such as C, C++ and FORTRAN.

You can use MATLAB in a wide range of applications, including signal and image processing, communications, control design, test and measurement, financial modeling and analysis, and computational biology. Add on tool boxes (collections of special purpose MATLAB functions, available separately) extend the MATLAB environment to solve particular classes of problems in these application areas.

MATLAB provides a number of features for documenting and sharing your work. You can integrate your MATLAB code with other languages and applications, and distribute your MATLAB algorithms and applications.

Features include:

- High level language for technical computing
- Development environment for managing code, files and data
- Interactive tools for iterative exploration, design and problem solving
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration
- 2-D and 3-D graphics functions for visualizing data
- Tools for building custom graphical user interfaces
- Functions of integrating MATLAB based algorithms with external applications and languages, such as C, C++, FORTRAN, COM, Java™ and Microsoft® Excel®

4.2.2 The MATLAB system

The MATLAB system consists of these main parts:

1) Desktop tools and development environment

This part of MATLAB is the set of tools and facilities that help you use and become more productive with MATLAB functions and files. Many of these tools are graphical user interfaces. It includes; the MATLAB desktop and command window, an editor and debugger, a code analyzer, and browsers for viewing help, the work space, and folders.

2) Mathematical function library

This library is a vast collection of computational algorithms ranging from elementary functions, like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigen values, Bessel functions, and fast Fourier transforms.

3) The language

The MATLAB language is a high level matrix/array language with control flow statements, functions, data structures, input/output, and object oriented programming features. It allows both "programming in the small" to rapidly create quick programs you do not intend to reuse. You can also do "programming in the large" to create complex application programs intended for reuse.

4) Graphics

MATLAB has extensive facilities for displaying vectors and matrices as graphs, as well as annotating and printing these graphs. It includes high level functions for 2-D and 3-D data visualization, image processing, animation, and presentation graphics. It also includes low level functions that allow you to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications.

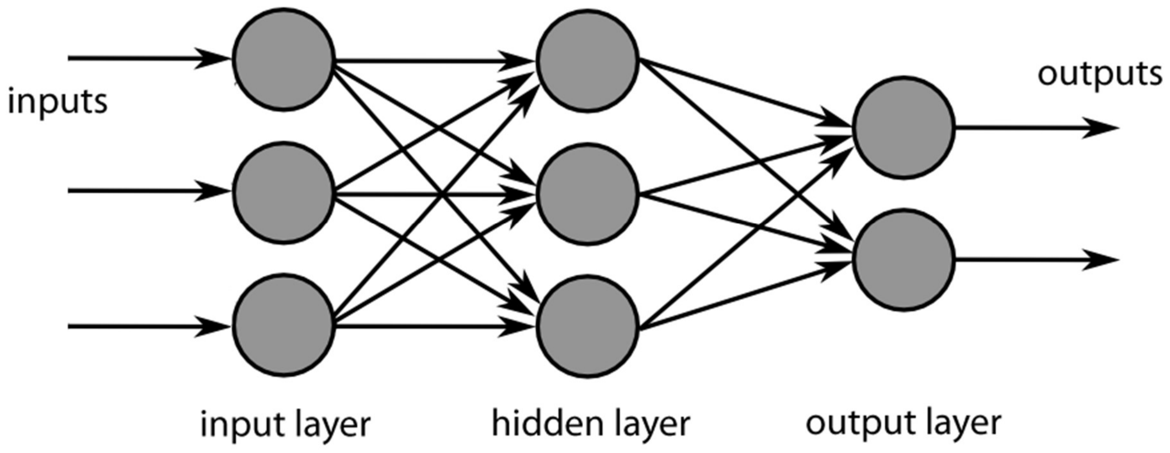
5) External interfaces

The external interfaces library allows you to write C/C++ and FORTRAN programs with MATLAB. It includes facilities for calling routines from

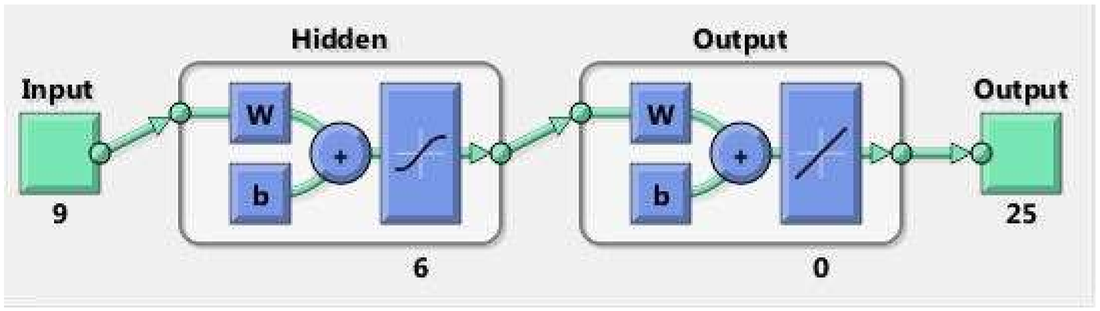
MATLAB (dynamic linking), for calling MATLAB as a computational engine, and for reading and writing MAT files.

4.3 Neural Network

The network we used for the program is feed forward network. An example of the feed forward network is given below.



The network we used in this project have 9 units in the input layer and 25 units in the output layer.



Feedforward networks consist of a series of layers. The first layer has a connection from the network input. Each subsequent layer has a connection from the previous layer. The final layer produces the network's output.

Feedforward networks can be used for any kind of input to output mapping. A feedforward network with one hidden layer and enough neurons in the hidden layers, can fit any finite input-output mapping problem.

MATLAB function for creation of network

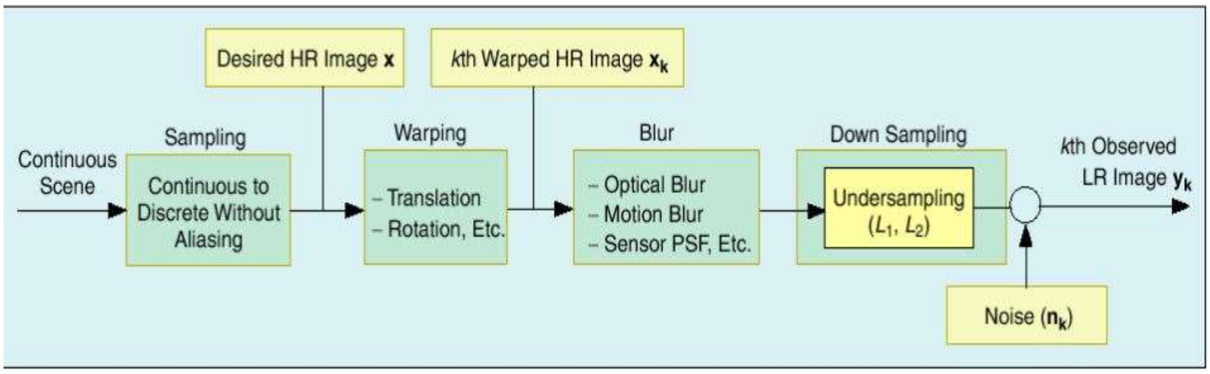
feedforwardnet(hiddenSizes,trainFcn)

hiddenSize : hidden layer sizes

trainFcn : Training function

Number of hidden layers we used for the network is 6. Because which give the maximum convergence. Training Function is 'traingdm'. The algorithm 'traingdm' uses back propagation algorithm.

4.4 Training set preparation



Desired HR Image $\mathbf{x}$
kth Warped HR Image $\mathbf{x}_k$
Continuous Scene
Sampling
Continuous to Discrete Without Aliasing
Warping
– Translation
– Rotation, Etc.
Blur
– Optical Blur
– Motion Blur
– Sensor PSF, Etc.
Down Sampling
Undersampling (L_1, L_2)
kth Observed LR Image $\mathbf{y}_k$
Noise ($\mathbf{n}_k$)

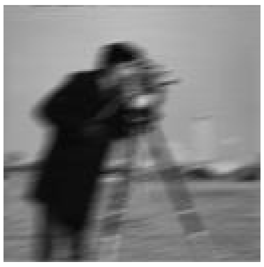
Figure 15: Image observation model

For the training of the neural network we selected different low resolution images and corresponding high resolution image. We choose the image cameraman.tif.

Figure 16: Cameaman.tif, Original 256x256 image

Training input set is prepared by down sampling, Shifting and blurring the original image. The first training input image is down sampled cameraman.tif. Resolution of the input image is 128x128. Training output image is original cameraman.tif. The second training input image is shifted and blurred 128x128 images. For this set the output image is the same as the first set.

Training set 1	
LR image (128x128)	**HR image** (256x256)

Training set 2	
LR Image 128x128,Blurred,Shifted	**HR Image** 256x256
	

Input and output matrix for training

To prepare the input matrix we patched the input image by 3x3 matrix with overlapping. For the output matrix patch matrix size is 5x5. These small matrix are converted to column matrix (i.e. 9x1 and 25x1) and concatenated together. Input matrix size is 9x16384 and output matrix size is 25x16384.

4.5 Training of network

traingdm

traingdm is a network training function that updates weight and bias values according to gradient descent with momentum. To train the neural network we use the Matlab code *net.trainFcn = 'traingdm'*

[net,tr,Y,E,Pf,Af] = train(net,P,T,Pi,Ai)

LHS

net: Network

P: Training input matrix

T: Training output matrix

Pi: Initial input delay conditions (default = zeros)

Ai: Initial layer delay conditions (default = zeros)

RHS

net :New network

tr: Training record (epoch and perf)

Y: Network outputs

E: Network errors

Pf: Final input delay conditions

Af: Final layer delay conditions

traingdm can train any network as long as its weight, net input, and transfer functions have derivative functions.

Backpropagation is used to calculate derivatives of performance *perf* with respect to the weight and bias variables X. Each variable is adjusted according to gradient descent with momentum,

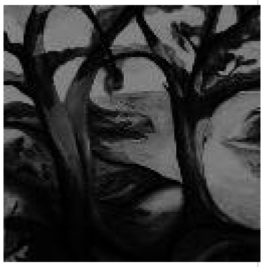
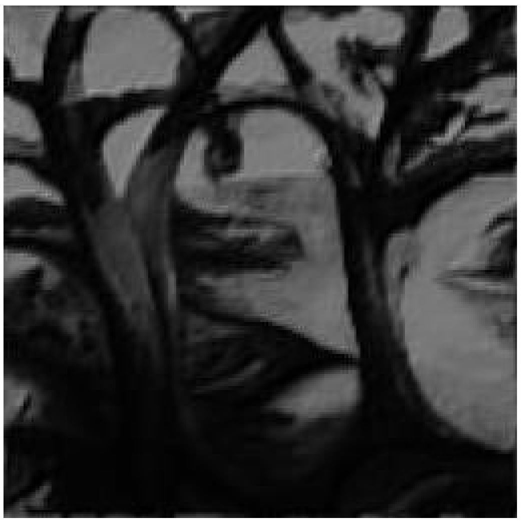
dX = mc*dXprev + lr*(1-mc)*dperf/dX

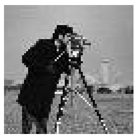
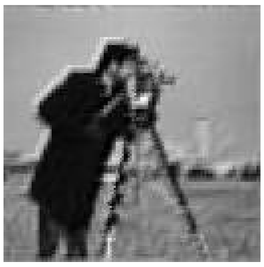
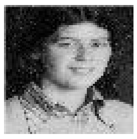
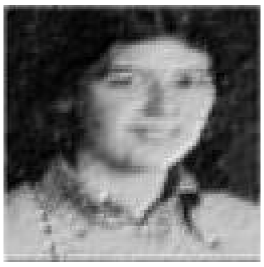
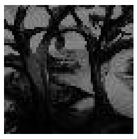
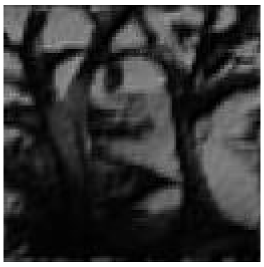
where dXprev is the previous change to the weight or bias.

Training stops when any of these conditions occurs:

- The maximum number of epochs (repetitions) is reached.
- The maximum amount of time is exceeded.
- Performance is minimized to the goal.
- The performance gradient falls below min_grad.
- Validation performance has increased more than max_fail times since the last time it decreased (when using validation).

4.6 Results

LR Image		HR Image		PSNR
128x128		256x256		28.18

LR Image		HR Image		PSNR
64x64		128x128		20.33
64x64		128x128		21.61
64x64		128x128		24.40

4.7 Conclusion and future scope

Super resolution is the process of obtaining a high resolution image from one or more low resolution images. We can use neural network for super resolution. Training of one image set will give rise to a generalized network for super resolution. Super resolution using neural network is very fast when comparing to other methods. In this project we used gray scale images for training and testing. But the same network can be used for color images also. When considering PSNR, the result is good. All the test result have PSNR greater than 20. With more training of the net work the PSNR will increase. Also taking DCT of the images is another way to increase PSNR.

Bibliography

[1] H. Greenspan, _Super-Resolution in Medical Imaging,_ The Computer Journal, vol. 52,no. 1, pp. 43_63, Jan. 2009.
[2] M. T. Merino and J. Nunez, _Super-Resolution of Remotely Sensed Images With Variable-Pixel Linear Reconstruction,_ IEEE Transactions on Geoscience and Remote Sensing, vol. 45, no. 5, pp. 1446_1457, May 2007.
[3] A. J. Tatem, H. G. Lewis, P. M. Atkinson, and M. S. Nixon, _Super-resolution target identication from remotely sensed images using a Hopfield neural network,_ IEEE Transactions
on Geoscience and Remote Sensing, vol. 39, no. 4, pp. 781_796, Apr. 2001.
[4] E. Engin and M. Özcan, _Moving target detection using super-resolution algorithms withan ultra wideband radar,_ International Journal of Imaging Systems and Technology,vol. 20, no. 3, pp. 237_244, 2010.
[5] S. Ebihara, M. Sato, and H. Niitsuma, _Super-Resolution of Coherent Targets by a Directional Borehole Radar,_ IEEE Transactions on Geoscience and Remote Sensing, vol. 38, no. 4, pp. 1725_1732, Jul. 2000.
[6] A. Gehani and J. H. Reif, _Super-Resolution Video Analysis for Forensic Investigations,_ in IFIP WG 11.9 International Conference on Digital Forensics, ser. IFIP, vol. 242. Springer, 2007, pp. 281_299.
[7] Y. Wang, R. Fevig, and R. R. Schultz, _Super-resolution mosaicking of UAV surveillance video,_ in IEEE International Conference on Image Processing (ICIP), 2008, pp. 345_348.
[8] T. S. Huang and R. Y. Tsai, _Multiframe image restoration and registration,_ Advances in Computer Vision and Image Processing, vol. 1, no. 7, pp. 317_339, 1984.
[9] M. Irani and S. Peleg, _Super resolution from image sequences,_ in 10th International Conference on Pattern Recognition, vol. 2, Jun. 1990, pp. 115_120.

[10] S. Borman and R. Stevenson, _Super-Resolution from Image Sequences _ A Review,_ in Midwest Symposium on Circuits and Systems, Notre Dame, IN, USA, 8 1998, pp. 374_378.

[11] S. C. Park, M. K. Park, and M. G. Kang, _Super-Resolution Image Reconstruction: A Technical Overview,_ IEEE Signal Processing Magazine, vol. 20, no. 3, pp. 21_36, 5 2003.

[12] C. Mancas-Thillou and M. Mirmehdi, _An Introduction to Super-Resolution Text,_ in Digital Document Processing, ser. Advances in Pattern Recognition. Springer London, 2007, pp. 305_327.

[13] J. Tian and K.-K. Ma, _A survey on super-resolution imaging,_ Signal, Image and Video Processing (SIViP), vol. 5, no. 3, pp. 329_342, 2011.

[14] H. Ur and D. Gross, _Improved Resolution from Subpixel Shifted Pictures,_ CVGIP:Graph. Models Image Process., vol. 54, no. 2, pp. 181_186, Mar. 1992.

[15] A. Papoulis, _Generalized Sampling Expansion,_ IEEE Transactions on Circuits and Systems,, vol. 24, no. 11, pp. 652_654, Nov. 1977.

[16] N. K. Bose and N. A. Ahuja, _Superresolution and Noise Filtering Using Moving Least Squares,_ IEEE Transactions on Image Processing, vol. 15, no. 8, pp. 2239_2248, Aug.2006.

[17] A. J. Patti, M. I. Sezan, and A. M. Tekalp, _Superresolution Video Reconstruction with Arbitrary Sampling Lattices and Nonzero Aperture Time,_ IEEE Transactions on Image Processing, vol. 6, no. 8, pp. 1064_1076, Aug. 1997.

[18] S. Rhee and M. G. Kang, _DCT-Based Regularized Algorithm for High-Resolution Image Reconstruction,_ in IEEE International Conference on Image Processing (ICIP). Los Alamitos, CA: IEEE, Oct. 1999, pp. 184_187.

[19] N. A. Woods, N. P. Galatsanos, and A. K. Katsaggelos, _Stochastic Methods for Joint Registration, Restoration, and Interpolation of Multiple

Undersampled Images,_ IEEE Transactions on Image Processing, vol. 15, no. 1, pp. 201_213, Jan. 2006.

[20] A. P. Dempster, N. M. Laird, and D. B. Rubin, _Maximum Likelihood from Incomplete Data via the EM Algorithm,_ Journal of the Royal Statistical Society, vol. 39, no. 1, pp. 1_38, 1977.

[21] N. Nguyen and P. Milanfar, _An E_cient Wavelet-Based Algorithm for Image Superresolution, in IEEE Internation Conference on Image Processing, 2000, pp. 351_354.

[22] S. E. El-Khamy, M. M. Hadhoud, M. I. Dessouky, B. M. Salam, and F. E. A. El-Samie,_Wavelet Fusion: a Tool to Break the Limits on LMMSE Image Super-Resolution,_ International Journal of Wavelets, Multiresolution and Information Processing (IJWMIP),vol. 4, no. 1, pp. 105_118, 2006.

[23] H. Ji and C. Fermüller, _Wavelet-Based Super-Resolution Reconstruction: Theory and Algorithm,_ in 9th European Conference on Computer Vision (ECCV), vol. 3954. Springer, 2006, pp. 295_307.

[24] _, _Robust Wavelet-Based Super-Resolution Reconstruction: Theory and Algorithm,in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 4, Apr.2009, pp. 649_660.

[25] M. B. Chappalli and N. K. Bose, _Simultaneous Noise Filtering and Super-Resolution With Second-Generation Wavelets,_ IEEE Signal Processing Letters, vol. 12, no. 11, pp.772_775, Nov. 2005.

[26] A. K. Katsaggelos, Digital Image Restoration. Heidelberg, Germany: Springer, 1991,vol. 23.

[27] R. C. Hardie, K. J. Barnard, and E. E. Armstrong, _Joint MAP registration and high resolution image estimation using a sequence of undersampled images,_ IEEE Transactions on Image Processing, vol. 6, no. 12, pp. 1621_1633, 1997.

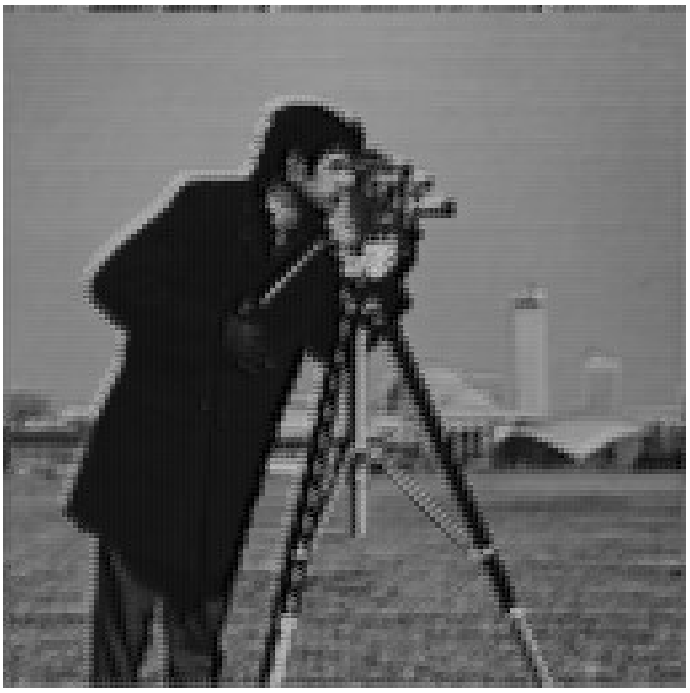
[28] J. Tian and K.-K. Ma, _Stochastic super-resolution image reconstruction,_ Jorunal of Visual Communication and Image Representation, vol. 21, no. 3, pp. 232_244, 2010.
[29] R. R. Schultz and R. L. Stevenson, _Extraction of High-Resolution Frames from Video Sequences,_ IEEE Transactions on Image Processing, vol. 5, no. 6, pp. 996_1011, Jun.1996.
[30] K. V. Suresh and A. N. Rajagopalan, _Robust and computationally e_cient superresolution algorithm,_ Journal of the Optical Society of America, vol. 24, no. 4, pp. 984_992,Apr. 2007.
[31] S. P. Belekos, N. P. Galatsanos, and A. K. Katsaggelos, _Maximum a Posteriori Video Super-Resolution Using a New Multichannel Image Prior,_ IEEE Transactions on Image Processing, vol. 19, no. 6, pp. 1451_1464, 2010.
[32] H. Shen, L. Zhang, B. Huang, and P. Li, _A MAP Approach for Joint Motion Estimation, Segmentation, and Super Resolution,_ IEEE Transactions on Image Processing, vol. 16, no. 2, pp. 479_490, Feb. 2007.
[33] B. C. Tom and A. K. Katsaggelos, _Reconstruction of a high-resolution image by simultaneous registration, restoration, and interpolation of low-resolution images,_ in IEEE International Conference on Image Processing (ICIP). IEEE, 1995, pp. 539_542.
[34] S. Baker and T. Kanade, _Limits on Super-Resolution and How to Break Them,_ IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 9, pp. 1167_1183,2002.

Results for multi frame super resolution

Blurred, shifted images of cameraman (Training Input)

Original cameraman (Training Output)

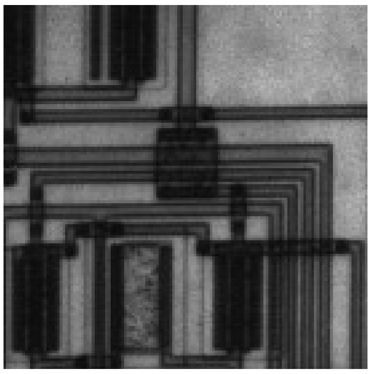
Input



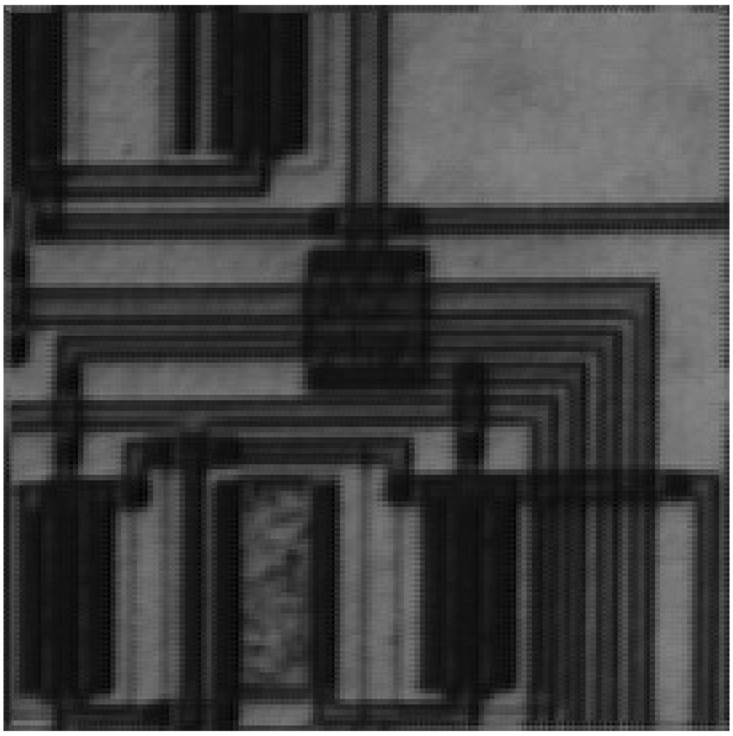
Output

PSNR 20.8118

Input



Output

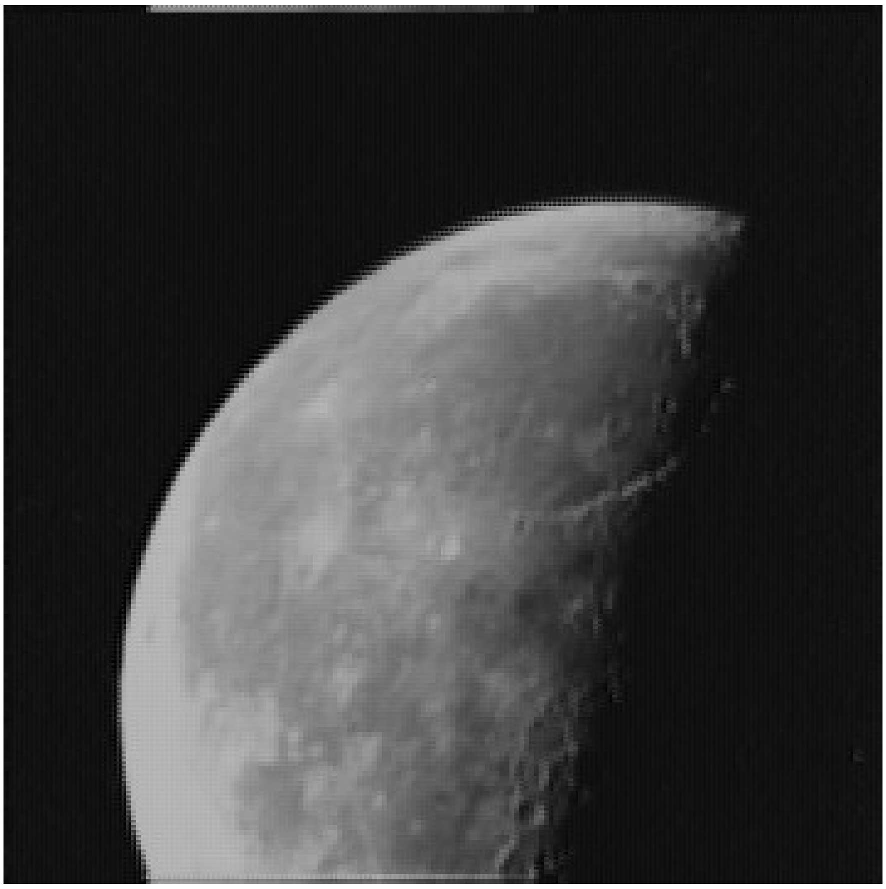


PSNR

21.4899

Input

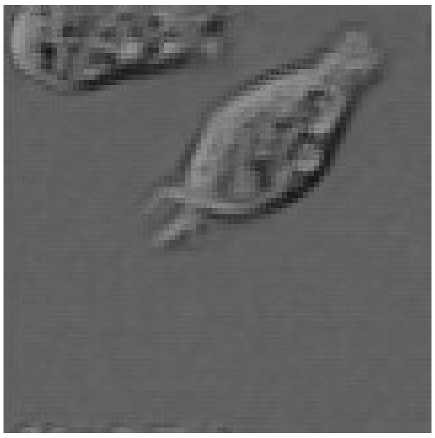
Output



PSNR 23.816

Input

Output



PSNR 24.7524