**CHAPTER 1**

**INTRODUCTION**

Inpresent times, the development of image processing is more which resulted in the increase of image quality. The high-resolution image is desirable for digital applications of two main categories that is, improvement of visual features for better perception of people and higher image perception for automated machines. The image with high resolution accounts for more image details and possess high frequency components.

Single picture super-resolution (SR), which focuses on recuperating a high-resolution picture from a single low-resolution picture is an old-style issue in Computer vision. This issue is naturally not well presented since an assortment of arrangements exist for some random low-resolution pixel. At the end of the day, it is an underdetermined backwards issue, of which arrangement isn't extraordinary. Such an issue is commonly alleviated by obliging the arrangement space by solid earlier data.

Single image super-resolution (SR), which aims at recovering a high-resolution image from a single low-resolution image, is a classical problem in computer vision. This problem is inherently ill-posed since a multiplicity of solutions exist for any given low-resolution pixel. In other words, it is an underdetermined inverse problem, of which solution is not unique. Such a problem is typically mitigated by constraining the solution space by strong prior information. To learn the prior, recent state-of-the-art methods mostly adopt the example-based strategy. These methods either exploit internal similarities of the same, or learn mapping functions from external low- and high-resolution exemplar pairs.

The external example-based methods can be formulated for generic image super-resolution, or can be designed to suit domain speciﬁc tasks, i.e., face hallucination, according to the training samples provided. The sparse-coding-based method is one of the representative external example-based SR methods. This method involves several steps in its solution pipeline. First, overlapping patches are densely cropped from the input image and pre-processed (e.g., subtracting mean and normalization). These patches are then encoded by a low-resolution dictionary. The sparse coefﬁcients are passed into a high-resolution dictionary for reconstructing high-resolution patches. The overlapping re constructed patches are aggregated (e.g., by weighted averaging) to produce the ﬁnal output. This pipeline is shared by most external example-based methods, which pay particular attention to learning and optimizing the dictionaries or building efﬁcient mapping functions. However, the rest of the steps in the pipeline have been rarely optimized or considered in an uniﬁed optimization framework.

In this project, we show that the aforementioned pipeline is equivalent to a deep convolutional neural network. Motivated by this fact, we consider a convolutional neural network that directly learns an end-to-end mapping between low- and high-resolution images. Our method differs fundamentally from existing external example-based approaches, in that ours does not explicitly learn the dictionaries or manifolds for modelling the patch space. These are implicitly achieved via hidden layers. Furthermore, the patch extraction and aggregation are also formulated as convolutional layers, so are involved in the optimization. In our method, the entire SR pipeline is fully obtained through learning, with little pre/postprocessing.

We name the proposed model Super-Resolution Convolutional Neural Network (SRCNN). The proposed SRCNN has several appealing properties. First, its structure is intentionally designed with simplicity in mind, and yet provides superior accuracy compared with state-of-the-art example-based methods

Overall, the contributions of this study are mainly in three aspects:

**I.** We present a fully convolutional neural network for image super-resolution. The network directly learns an end-to-end mapping between low and high-resolution images, with little pre/postprocessing beyond the optimization.

**II.** We establish a relationship between our deep learning learning-based SR method and the traditional sparse-coding-based SR methods. This relationship provides a guidance for the design of the network structure.

**III.** We demonstrate that deep learning is useful in the classical computer vision problem of super resolution, and can achieve good quality and speed.

The present adds to the initial version in signiﬁcant way. Firstly, we improve the SRCNN by introducing larger ﬁlter size in the non-linear mapping layer, and explore deeper structures by adding nonlinear mapping layers. Secondly, we extend the SRCNN to process three colour channels (either in YCbCr or RGB colour space) simultaneously. Experimentally, we demonstrate that performance can be improved in comparison to the single-channel network. Thirdly, considerable new analyses and intuitive explanations are added to the initial results.

In case of satellite imagery, the hardware required for imaging is quite expensive so having computing software which can improve the quality of the image will be a huge gain as the hardware can be compromised over computational costs (which are usually much less compared to an upgrade in the hardware). The image processing technique used to obtain high quality image from a degraded image is Super-Resolution. The single image super resolution (SISR) technique deals with conversion of single low-resolution image to a single high-resolution image. SISR is considered challenging as it does not have the high frequency image components to reconstruct the high-resolution image from low-resolution image. Another problem faced by the SISR is that a low-resolution image can have many different possible high-resolution outcomes. So, the image quality is usually limited.

**CHAPTER-2**

**LITERATURE SURVEY**

This study proposes a novel super resolution model based on adaptive sparse representation and self-learning frameworks. The super resolution method is to take more samples of the scene so as to get some extra information which can be used, while merging the samples to get a high-resolution image. These samples can be acquired by sub-pixel shifts, by changing the amount of blur. HR means that pixel density within an image is high, and therefore an HR image can offer more details that are important in many applications, the major advantage of the super resolution approach is that it may cost less and the existing LR imaging systems can be still utilized. Synthetic zooming of region of interest (ROI) is another important application in surveillance, forensic, scientific, medical, and satellite. This application is most suitable for magnifying objects in the scene such as the face of a criminal or the license plate of a car. This study proposes a novel super-resolution regularization model based on adaptive sparse learning frameworks. The adaptive sparsity regularization term constrains the reconstructed image with an adaptive sparse representation, which successfully harmonizes the sparse representation and the collaborative representation adaptively via producing suitable coefficients. To construct a more effective dictionary, the high-frequency features from the underlying image patches are extracted, and the dictionary learning and sparse representation are integrated. To this end, the alternating minimization algorithm is used to divide this model into three sub-problems, and the alternating direction method of multipliers and iterative back-projection method are used to solve the sub-problems.

To illustrate the effectiveness of the proposed method, additional experiments are conducted on some generic images. Compared with some state-of-the-art algorithms, the experimental results demonstrate that the proposed method achieves better results in terms of both visual quality and noise immunity. Traditional works have shown that patches in a natural image tend to redundantly recur many times inside the image, both within the same scale, as well as across different scales. Make full use of these multi-scale information can improve the image restoration performance.

However, the current proposed deep learning-based restoration methods do not take the multi-scale information into account. In this paper, we propose a dilated convolution-based inception module to learn multi-scale information and design a deep network for single image super resolution. Different dilated convolution learns different scale feature, then the inception module concatenates all these features to fuse multi-scale information. In order to increase the reception field of our network to catch more contextual information, we cascade multiple inception modules to constitute a deep network to conduct single image super resolution. Y. Sun Et-al, presented a compressive sensing based on a redundant dictionary has been successfully applied in super resolution imaging. However, due to the neglect of the local and nonlocal interactions of patches of a single image, the reconstructed results are not satisfactory in noise suppression and edge sharpness. In this paper, we propose an improved method by adding steering kernel regression and a nonlocal means filter as two regularization terms and use an efficient clustering sub-dictionary learning scheme. We further demonstrate better results on true images in terms of traditional image quality assessment metrics. Sun Et-al, presented an observation for medical imaging and astronomical, high-resolution (HR) images are urgently desired and required. In recent years, many researchers have proposed various ways to achieve the goal of image super-resolution (SR), ranging from simple linear interpolation schemes to nonlinear complex methods.

We deal with the SR reconstruction problem based on the theory of compressive sensing, which uses a redundant dictionary instead of a conventional orthogonal basis. We further demonstrate better results on true images in terms of peak signal-to-noise ratio (PSNR) and root mean-square error (RMSE) and give several important improvements, compared with other methods. Agrawal et-al, presented a technique for enhancing resolution of images by interpolating high frequency sub-bands generated using lifting wavelet transform (LWT) and spatial information of input low resolution (LR) image. Stationary wavelet transforms (SWT) is used at intermediate stage for edge enhancement. The input image is decomposed using LWT in order to generate high frequency (HF) sub bands. The generated HF sub bands are interpolated further. Different high frequency sub-bands obtained through SWT are added to correct the estimated HF sub bands. The input LR image is interpolated in parallel. All these sub-bands and estimated LR image are reconstructed by inverse lifting wavelet transformation (ILWT) to produce high resolution image. The qualitative, quantitative and visual images of the described technique show the superiority of the proposed method over conventional and state-of-the art methods. H. Ashikaga et-al, presented Single-image super resolution is a process of obtaining a high-resolution image from a set of low-resolution observations by signal processing.

While super resolution has been demonstrated to improve image quality in scaled down images in the image domain, its effects on the Fourier-based image acquisition technique, such as MRI, remains unknown. We performed high resolution ex vivo late gadolinium enhancement (LGE) magnetic resonance imaging (0.4 × 0.4 × 0.4 mm3) in position fraction swine hearts (nD24). The swine hearts were divided into the training set low resolution images were simulated from the high-resolution images. In the training set, super resolution dictionaries with pairs of small matching patches of the high and low-resolution images were created. In the test set, super resolution recovered high resolution images from low resolution images using the dictionaries. The same algorithm was also applied to patient LGE (nD4) to assess its effects. Compared with interpolated images, super resolution significantly improved basic image quality indices (P < 0.001). Super resolution using Fourier-based zero padding achieved the best image quality. However, the magnitude of improvement was small in images with zero padding. Super resolution substantially improved the spatial resolution of the patient LGE images by sharpening the edges of the heart and the scar. E. Faramarzi et-al, presented, a unified blind method for multi-image super resolution (MISR or SR), single-image blur deconvolution (SIBD), and multi-image blur deconvolution (MIBD) of low resolution (LR) images degraded by linear space invariant (LSI) blur, aliasing, and additive white Gaussian noise (AWGN). The proposed approach is based on alternating minimization (AM) of a new cost function with respect to the unknown high-resolution (HR) image and blurs. The regularization term for the HR image is based upon the Huber Markov random field (HMRF) model, which is a type of variational integral that exploits the piecewise smooth nature of the HR image. The blur estimation process is supported by an edge emphasizing smoothing operation, which improves the quality of blur estimates by enhancing strong soft edges toward step edges, while filtering out weak structures. The parameters are updated gradually so that the number of salient edges used for blur estimation increases at each iteration.

For better performance, the blur estimation is done in the filter domain rather than the pixel domain i.e., using the gradients of the LR and HR images.

The regularization term for the blur is Gaussian (L2 norm), which allows for fast non iterative optimization in the frequency domain. We accelerate the processing time of SR reconstruction by separating the up sampling and registration processes from the optimization procedure. Simulation results on both synthetic and real-life images (from a novel computational imager) confirm the robustness and effectiveness of the proposed method.

**CHAPTER 3**

**SYSTEM ANALYSIS**

Single picture super-resolution (SR), which focuses on recuperating a high-resolution picture from a single low-resolution picture is an old-style issue in Computer vision. This issue is naturally not well presented since an assortment of arrangements exist for some random low-resolution pixel. At the end of the day, it is an underdetermined backwards issue, of which arrangement isn't extraordinary. Such an issue is commonly alleviated by obliging the arrangement space by solid earlier data.

**3.1 EXISTING SYSTEM**

Image Super resolution has been implemented in several different ways using Technologies like MATLAB. The system shows an acceptable accuracy for image super resolution by implementing Neural Networks in the system.

**DISADVANTAGES**

* The existing systems are implemented on MATLAB and hence are not opensource.
* It takes up more resources and overall gives less accuracy.

**3.2 PROPOSED SYSTEM**

In our proposes system, Python based Deep learning algorithms are being implemented for making Super Resolution images using a Low-resolution image. The System is using Convolutional Neural Network for doing the task.

**ADVANTAGES**

* Usage of Machine Learning Algorithm makes the system more reliable and accurate.
* Convolutional Neural Network algorithm is implemented.

The authors of the SRCNN describe their network, pointing out the equivalence of their method to the sparse-coding method4, which is a widely used learning method for image SR. This is an important and educational aspect of their work, because it shows how example-based learning methods can be adapted and generalized to CNN models.

The SRCNN consists of the following operations:

* 1. **Pre-processing**: Up-scales LR image to desired HR size.
  2. **Feature extraction**: Extracts a set of feature maps from the up-scaled LR image.
  3. **Non-linear mapping**: Maps the feature maps representing LR to HR patches.
  4. **Reconstruction**: Produces the HR image from HR patches.

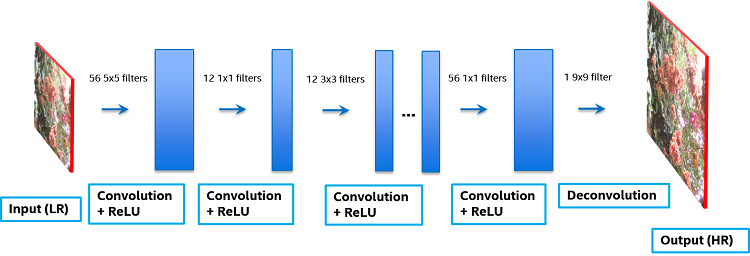


Fig. 3.1. system flow

As the title suggests, the SRCNN is a deep convolutional neural network that learns the end-to-end mapping of low-resolution to high-resolution images. As a result, we can use it to improve the image quality of low-resolution images. To evaluate the performance of this network, we will be using three image quality metrics: peak signal to noise ratio (PSNR), mean squared error (MSE), and the structural similarity (SSIM) index.

During this project, you will learn how to:

* + use the PSNR, MSE, and SSIM image quality metrics,
  + process images using OpenCV,
  + convert between the RGB, BGR, and YCrCb color spaces,
  + build deep neural networks in Keras,
  + deploy and evaluate the SRCNN network

**3.3 FEASIBILITY STUDY**

The first and foremost strategy for development of a project starts from the thought of designing a mail enabled platform for a small firm in which it is easy and convenient of sending and receiving messages, there is a search engine, address book and also including some entertaining games. When it is approved by the organization and our project guide the first activity, ie., preliminary investigation begins. The activity has three parts:

• Request Clarification

• Feasibility Study

• Request

After the approval of the request to the organization and project guide, with an investigation being considered, the project request must be examined to determine precisely what the system requires. Here our project is basically meant for users within the company whose systems can be interconnected by the Local Area Network (LAN). In today’s busy schedule man need everything should be provided in a readymade manner. So, taking into consideration of the vastly use of the net in day to day life, the corresponding development of the portal came into existence.

**3.4 FEASIBILITY ANALYSIS**

An important outcome of preliminary investigation is the determination that the system request is feasible. This is possible only if it is feasible within limited resource and time. The different feasibilities that have to be analyzedare

**3.4.1 Operational Feasibility**

Operational Feasibility deals with the study of prospects of the system to be developed. This system operationally eliminates all the tensions of the Admin and helps him in effectively tracking the project progress. This kind of automation will surely reduce the time and energy, which previously consumed in manual work. Based on the study, the system is proved to be operationally feasible.

**3.4.2 Economic Feasibility**

Economic Feasibility or Cost-benefit is an assessment of the economic justification for a computer-based project. As hardware was installed from the beginning & for lots of purposes thus the cost on project of hardware is low. Since the system is a network based, any number ofemployeesconnectedtotheLANwithinthatorganizationcanusethistoolfromat anytime. TheVirtualPrivateNetworkistobedevelopedusingtheexistingresourcesoftheorganization. So the project is economically feasible.

**3.4.3 Technical Feasibility**

According to Roger S. Pressman, Technical Feasibility is the assessment of the technical resources of the organization. The organization needs IBM compatible machines with a graphical web browser connected to the Internet and Intranet. The system is developed for platform Independent environment. Java Server Pages, JavaScript, HTML, SQL server and WebLogic Server are used to develop the system. The technical feasibility has been carried out. The system is technically feasible for development and can be developed with the existing facility

**CHAPTER 4**

**SOFTWARE REQUIREMENT ANALYSIS**

To solve actual problems in an industry setting, a software engineers or a team of engineers must incorporate a development strategy that encompasses that process, methods, and tools. This strategy is often referred to as process model or a software engineering paradigm. a process model for software engineering is chosen based on the nature of the project and application.

**4.1 SOFTWARE REQUIREMENTS SPECIFICATION (SRS)**

**A software requirements specification** (SRS) is a detailed description of a software system to be developed with its functional and non-functional requirements. The SRS is developed based the agreement between customer and contractors. It may include the use cases of how user is going to interact with software system. The software requirement specification document consistent of all necessary requirements required for project development. To develop the software system, we should have clear understanding of Software system. To achieve this, we need to continuous communication with customers to gather all requirements.

A good SRS defines the how Software System will interact with all internal modules, hardware, communication with other programs and human user interactions with wide range of real-life scenarios. Using theSoftware requirements specification(SRS) document on QA lead, managers create test plan. It is very important that testers must be cleared with every detail specified in this document in order to avoid faults in test cases and its expected results.

It is highly recommended to review or test SRS documents before start writing test cases and making any plan for testing. Let’s see how to test SRS and the important point to keep in mind while testing it.

**1**. Correctness of SRS should be checked: Since the whole testing phase is dependent on SRS, it is very important to check its correctness. There are some standards with which we can compare and verify.

**2**. Ambiguity should be avoided. Sometimes in SRS, some words have more than one meaning and this might confused testers making it difficult to get the exact reference. It is advisable to check for such ambiguous words and make the meaning clear for better understanding.

**3**. Requirements should be complete. When tester writes test cases, what exactly is required from the application, is the first thing which needs to be clear. For e.g. if application needs to send the specific data of some specific size then it should be clearly mentioned in SRS that how much data and what is the size limit to send.

**4**. Consistent requirements. The SRS should be consistent within itself and consistent to its reference documents. If you call an input “Start and Stop” in one place, don’t call it “Start/Stop” in another. This sets the standard and should be followed throughout the testing phase.

**5**. Verification of expected result: SRS should not have statements like “Work as expected”, it should be clearly stated that what is expected since different testers would have different thinking aspects and may draw different results from this statement.

**6**. Testing environment: some applications need specific conditions to test and also a particular environment for accurate result. SRS should have clear documentation on what type of environment is needed to set up.

**7**. Pre-conditions defined clearly: one of the most important part of test cases is pre-conditions. If they are not met properly then actual result will always be different expected result. Verify that in SRS, all the pre-conditions are mentioned clearly.

**8**. Requirements ID: these are the base of test case template. Based on requirement Ids, test case ids are written. Also, requirements ids make it easy to categorize modules so just by looking at them, tester will know which module to refer. SRS must have them such as id defines a particular module.

**9**. Security and Performance criteria: security is priority when a software is tested especially when it is built in such a way that it contains some crucial information when leaked can cause harm to business. Tester should check that all the security related requirements are properly defined and are clear to him. Also, when we talk about performance of a software, it plays a very important role in business so all the requirements related to performance must be clear to the tester and he must also know when and how much stress or load testing should be done to test the performance.

**10**. Assumption should be avoided: sometimes when requirement is not cleared to tester, he tends to make some assumptions related to it, which is not a right way to do testing as assumptions could go wrong and hence, test results may vary. It is better to avoid assumptions and ask clients about all the “missing requirements” to have a better understanding of expected results.

**11**. Deletion of irrelevant requirements: there are more than one team who work on SRS so it might be possible that some irrelevant requirements are included in SRS. Based on the understanding of the software, tester can find out which are these requirements and remove them to avoid confusions and reduce work load.

**12**. Freeze requirements: when an ambiguous or incomplete requirement is sent to client to analyse and tester gets a reply, that requirement result will be updated in the next SRS version and client will freeze that requirement. Freezing here means that result will not change again until and unless some major addition or modification is introduced in the software.

**4.2 SYSTEM REQUIREMENTS**

System requirements are the required specifications a device must have in order to use certain [hardware](https://techterms.com/definition/hardware) or [software](https://techterms.com/definition/software). For example, a computer may require a specific I/O [port](https://techterms.com/definition/port) to work with a [peripheral device](https://techterms.com/definition/peripheral). A smartphone may need a specific [operating system](https://techterms.com/definition/operating_system) to run a particular [app](https://techterms.com/definition/app)lication.

**4.2.1 Software Requirements**

Operating system : Windows10

Programming language : Python3.6

**4.2.2 Hardware Requirements**

Processor : i5

Hard Disk : 500MB (minimum)

RAM : 8GB

**CHAPTER 5**

**CONVOLUTIONAL NEURAL NETWORKS FOR**

**IMAGE SUPER-RESOLUTION**

According to the image priors, single-image super resolution algorithms can be categorized into four types – prediction models, edge-based methods, image statistical methods and patch based (or example-based) methods. These methods have been thoroughly investigated and evaluated in Yang et al.’s work. Among them, the example-based methods achieve the state-of-the-art performance. The internal example-based methods exploit the self-similarity property and generate exemplar patches from the input image. It is ﬁrst proposed in Glasner’s work, and several improved variants are proposed to accelerate the implementation. The external example-based methods, learn a mapping between low/high resolution patches from external datasets. These studies vary on how to learn a compact dictionary or manifold space to relate low/high-resolution patches, and on how representation schemes can be conducted in such spaces. In the pioneer work of Freeman et al, the dictionaries are directly presented as low/high-resolution patch pairs, and the nearest neighbor (NN) of the input patch is found in the low-resolution space, with its corresponding high-resolution patch used for reconstruction. Chang et al, introduce a manifold embedding technique as an alternative to the NN strategy. In Yang et al.’s work, the above NN correspondence advances to a more sophisticated sparse coding formulation. Other mapping functions such as kernel regression, simple function, random forest and anchored neighborhood regression are proposed to further improve the mapping accuracy and speed. The sparse coding-based method and its several improvements are among the state-of-the-art SR methods nowadays. In these methods, the patches are the focus of the optimization; the patch extraction and aggregation steps are considered as pre/post-processing and handled separately. The majority of SR focus on gray-scale or single-channel image super-resolution. For color images, the aforementioned methods ﬁrst transform the problem to a different color space (YCbCr or YUV), and SR is applied only on the luminance channel. There are also works attempting to super-resolve all channels simultaneously. For example, Kim and Kwon and Dai et al, apply their model to each RGB channel and combined them to produce the ﬁnal results.

However, none of them has analyzed the SR performance of different channels, and the necessity of recovering all three channels.

**5.1 CONVOLUTIONAL NEURAL NETWORKS**

Convolutional neural networks (CNN) date back decades and deep CNNs have recently shown an explosive popularity partially due to its success in image classiﬁcation. They have also been successfully applied to other computer vision ﬁelds, such as object detection, face recognition, and pedestrian detection. Several factors are of central importance in this progress:

1. the efﬁcient training implementation on modern powerful GPUs
2. the proposal of the Rectiﬁed Linear Unit (ReLU) which makes convergence much faster while still presents good quality, and
3. the easy access to an abundance of data (like ImageNet) for training larger models. Our method also beneﬁts from these progresses.

**5.2 DEEP LEARNING FOR IMAGE RESTORATION**

There have been a few studies of using deep learning techniques for image restoration. The multi-layer perceptron (MLP), whose all layers are fully-connected (in contrast to convolutional), is applied for natural image denoising and post-deblurring denoising. More closely related to our work, the convolutional neural network is applied for natural image denoising and removing noisy patterns (dirt/rain). These restoration problems are more or less denoising-driven. Cui et al. propose to embed auto-encoder networks in their super resolution pipeline under the notion internal example-based approach. The deep model is not speciﬁcally designed to be an end-to-end solution, since each layer of the cascade requires independent optimization of the self-similarity search process and the auto-encoder. On the contrary, the proposed SRCNN optimizes an end-to end mapping. Further, the SRCNN is faster at speed. It is not only a quantitatively superior method, but also a practically useful one.

**5.3 EXISTING TECHNIQUES FOR SUPER IMAGE RESOLUTION**

The super-resolution of a given single low-resolution input image can be achieved through various techniques, which are better than the other in terms of their individual performances. In this section we will briefly summarize their principles. Based on the application point of view single-image super-resolution approaches can be classified into three major categories: – Interpolation-Based, Reconstruction-Based, and Example-Based. Fig. 5.1 depicts this classification.

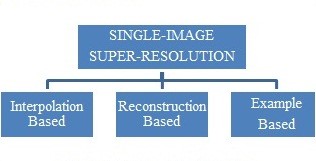


Fig. 5.1. approaches of single-image SR

# 5.3.1. Interpolation-Based Super-Resolution

# The simplest way to provide super-resolution is to apply interpolation on the sampled visual data acquired from the sensor. It is one of the early super-resolution algorithms based on resampling (a mathematical technique used to create a new version of the image with a different width and/or height in pixels). This approach, for example, which is present in digital cameras via the digital zoom, ultimately relies on the operations based on linear filtering.

This technique has advantage of less computational complexity due to its simplicity and also real-time applications are possible. The low-frequency (LF) band of the high-resolution image will be more or less accurately reconstructed.

However there are some limitations like, it is not possible to obtain the high-frequency information in the resized image due to low-pass behaviour of interpolation filters (bilinear, bicubic, Lanczos, it is unable to find the missing spectral contents of the resampled image, and the results often appear over-smooth (softer) and have jagged (rough) artifacts along the edges, which can be overcome in example-based SR.

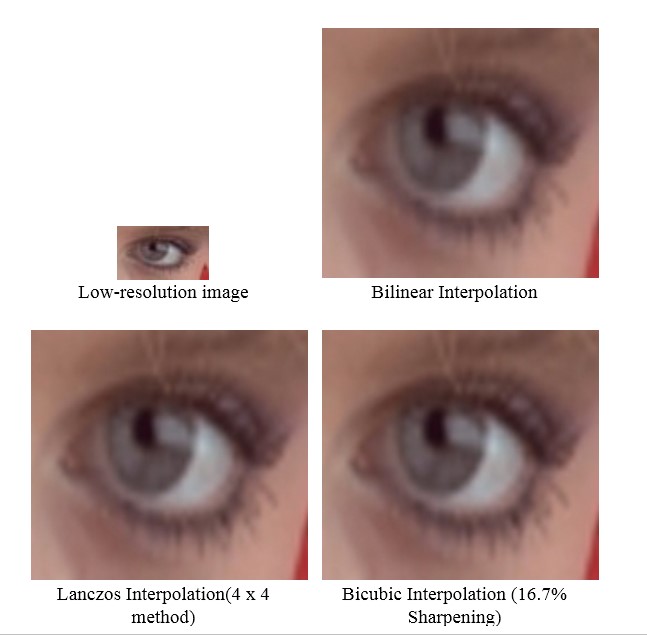


Fig 5.2. Behaviour of different interpolation filters.

# 5.3.2 Reconstruction-Based Super-Resolution

The reconstruction-based super-resolution approaches apply various smoothness priors and impose the constraint that when properly down sampled, the high-resolution image should reproduce the original low-resolution image.

The reconstruction-based super-resolution approaches are formulated under the framework of enforcing a reconstruction constraint and imposing a prior knowledge on high-resolution images. The reconstruction constraint requires that the high-resolution image via smoothing and down-sampling should be as close as possible to the low-resolution image.

In these methods, the imposed prior is not true for arbitrary images and many ringing artifacts may appear in the high-resolution image, which is one of the limitations of the reconstruction-based super-resolution technique.

# 5.3.3 Example-Based Super-Resolution

The second, and arguably one of the most successful, alternative is to estimate the high-resolution version of a low-resolution single-image by exploiting examples. The algorithms of example-based super-resolution problems are based on machine learning models exploiting available examples.

For example, we are given a low-resolution single image of a landscape with the lower part of the image showing the sea and the upper part showing the blue sky with some clouds. If we are having a database of high-resolution sea and sky images, which we can, for example, downscale to the resolution of our visual input, look for the low-resolution examples that best resembles or matches the two parts of our given image and reconstruct the corresponding high-resolution version of the given image with a montage (technique of selecting, editing and piecing together separate sections of film to form a continuous whole) of the sea and sky high resolution parts.

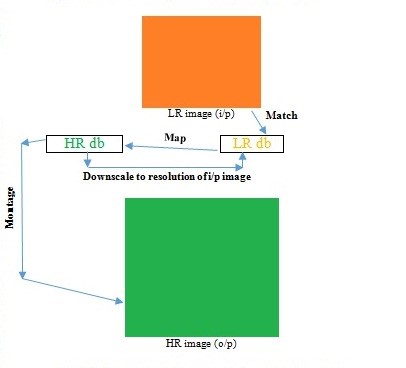


fig. 5.3. example-based super-resolution

The example-based super-resolution technique can be broadly classified into two categories parametric and non-parametric methods. The non-parametric methods can be further classified into internal learning and external learning methods

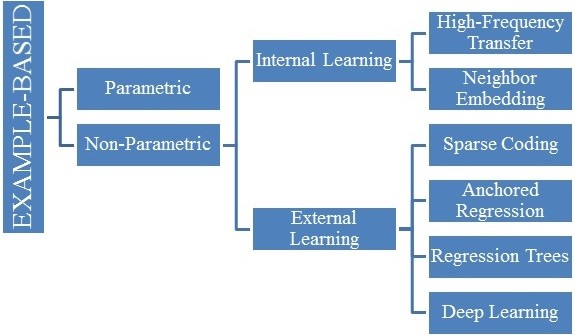


Fig. 5.4. classification of example-based super-resolution

However, there are some limitations like, due to insufficient training examples, high-frequency artifacts often appear in the result, rely on training set.

# 5.3.4 Parametric Vs Non-Parametric Methods

# Parametric super-resolution methods model the super-resolution problem with mapping functions which are controlled by a relatively compact amount of parameters. These parameters are learned from the examples that do not necessarily come from the input image. Parametric models are comparatively more powerful than non-parametric models in terms of efficiency, as, less data are needed to estimate the models.

Non-parametric models differ from the parametric models in that the model structure is not specified a priori. It is directly determined from the available training data. The term non-parametric does not imply that they completely lack parameters, but rather that the number and nature of the parameters are flexible and can depend upon the training data. The non-parametric models have a very desirable property of making fewer assumptions. The particular feature of a nonparametric model is that the input low-resolution image is decomposed into overlapping patches, and the reconstructed high-resolution image is obtained by combining the contributions of the computed overlapping output patches.

# 5.3.5 Internal Vs External Learning

The main idea behind the methods under internal learning category is that the patch examples are obtained directly from the input image (i.e., examples are internal, extracted from the input image), exploiting the cross-scale self-similarity property of natural images. This learning method has advantages of implicit adaptivity to the image contents, robust to noise, and can able to obtain better upscale results (including sharper edges and better preservation of texture). However, there are some limitations like, this method of learning requires iterative application for providing large up-scaling factors, the cross-scale self-similarity property of natural images degrades with large scale differences, and the computational cost is high.

**CHAPTER 6**

**SYSTEM ARCHITECTURE**

**6.1 FORMULATION**

Consider a single low-resolution image, we ﬁrst upscale it to the desired size using bicubic interpolation, which is the only pre-processing we perform3. Let us denote the interpolated image as Y. Our goal is to recover from Y an image F(Y) that is as similar as possible to the ground truth high-resolution image X. For the ease of presentation, we still call Y a “low-resolution” image, although it has the same size as X. We wish to learn a mapping F, which conceptually consists of three operations:

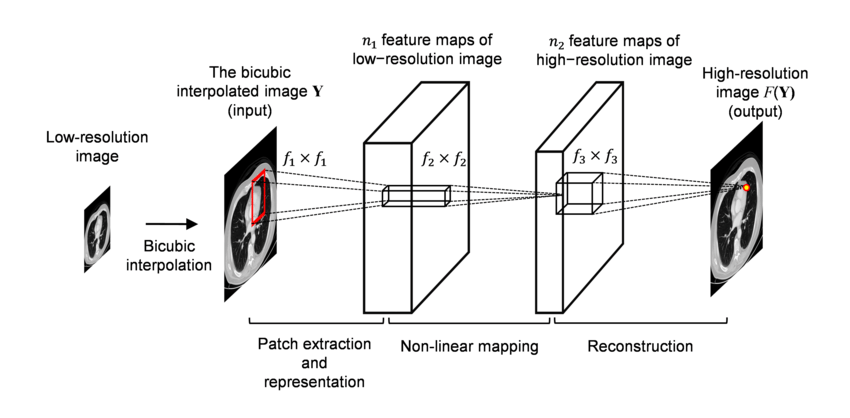


Fig. 6.1. SRCNN Architecture

* + 1. **Patch Extraction and Representation**

This operation extracts (overlapping) patches from the low-resolution image Y and represents each patch as a high-dimensional vector. These vectors comprise a set of feature maps, of which the number equals to the dimensionality of the vectors.

A popular strategy in image restoration is to densely extract patches and then represent them by a set of pre-trained bases such as PCA, DCT, Haar, etc. This is equivalent to convolving the image by a set of ﬁlters, each of which is a basis. In our formulation, we involve the optimization of these bases into the optimization of the network. Formally, our ﬁrst layer is expressed as an operation

F1: F1(Y) = max (0, W1 ∗Y + B)

where W1 and B1 represent the ﬁlters and biases respectively, and’∗’ denotes the convolution operation. Here, W1 corresponds to n1 ﬁlters of support c×f1×f1, where c is the number of channels in the input image, f1 is the spatial size of a ﬁlter. Intuitively, W1 applies n1 convolutions on the image, and each convolution has a kernel size c × f1 × f1. The output is composed of n1 feature maps. B1 is an n1-dimensional vector, whose each element is associated with a ﬁlter. We apply the Rectiﬁed Linear Unit (ReLU, max (0, x))on the ﬁlter responses.

**6.1.2 Non-Linear Mapping**

This operation nonlinearly maps each high-dimensional vector onto another high-dimensional vector. Each mapped vector is conceptually their presentation of a high-resolution patch. These vectors comprise another set of feature map.

The ﬁrst layer extracts an n1-dimensional feature for each patch. In the second operation, we map each of these n1-dimensional vectors into an n2-dimensional one. This is equivalent to applying n2 ﬁlters which have a trivial spatial support 1×1. This interpretation is only valid for 1×1 ﬁlters. But it is easy to generalize to larger ﬁlters like 3 × 3 or 5 × 5. In that case, the non-linear mapping is not on a patch of the input image instead, it is on a 3×3 or 5×5 “patch” of the feature map. The operation of the second layer

F2(Y) = max (0, W2 ∗F1(Y) + B2)

Here W2 contains n2 ﬁlters of size n1×f2×f2, and B2 is n2-dimensional. Each of the output n2-dimensional vectors is conceptually a representation of a high-resolution patch that will be used for reconstruction. It is possible to add more convolutional layers to increase the non-linearity. But this can increase the complexity of the model (n2 × f2 × f2 × n2 parameters for one layer), and thus demands more training time. We will explore deeper structures by introducing additional non-linear mapping layers in Section.

**6.1.3 Reconstruction**

This operation aggregates the above high-resolution patch-wise representations to generate the ﬁnal high-resolution image. This image is expected to be similar to the ground truth X.

In the traditional methods, the predicted overlapping high-resolution patches are often averaged to produce the ﬁnal full image. The averaging can be considered as a pre-deﬁned ﬁlter on a set of feature maps (where each position is the “ﬂattened” vector form of a high-resolution patch). Motivated by this, we deﬁne a convolutional layer to produce the ﬁnal high-resolution image:

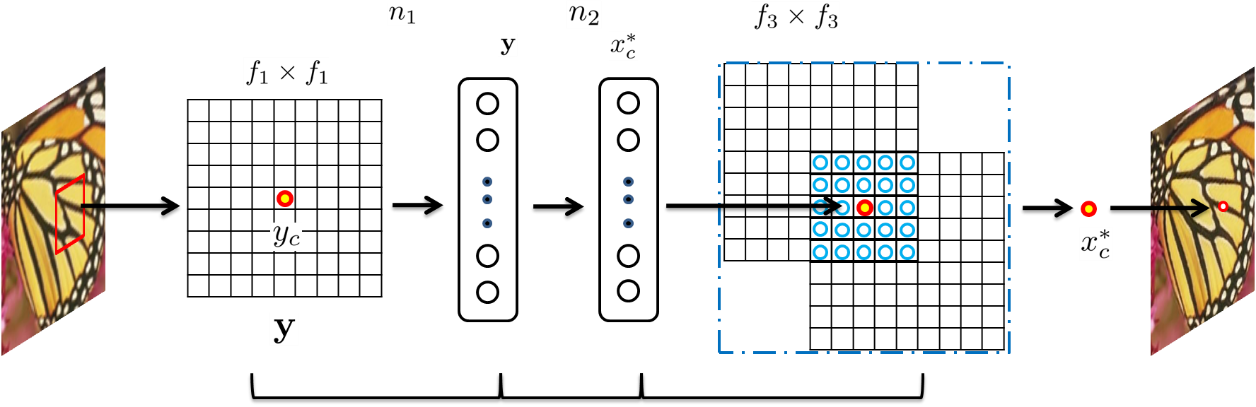
F(Y) = W3 ∗F2(Y) + B3.

Here W3 corresponds to c ﬁlters of a size n2 ×f3 ×f3, and B3 is a c-dimensional vector. If the representations of the high-resolution patches are in the image domain (i.e., we can simply reshape each representation to form the patch), we expect that the ﬁlters act like an averaging ﬁlter; if the representations of the high-resolution patches are in some other domains (e.g. Coefﬁcients in terms of some bases), we expect that W3 behaves like ﬁrst projecting the coefﬁcients onto the image domain and then averaging. In either way, W3 is a set of linear ﬁlters.

Interestingly, although the above three operations are motivated by different intuitions, they all lead to the same form as a convolutional layer. We put all three operations together and form a convolutional neural network. In this model, all the ﬁltering weights and biases are to be optimized. Despite the succinctness of the overall structure, our SRCNN model is carefully developed by drawing extensive experience resulted from signiﬁcant progresses in super-resolution.

* 1. **RELATIONSHIP TO SPARSE-CODING-BASED METHODS**

We show that the sparse-coding-based SR methods can be viewed as a convolutional neural network. Figure 6.2 shows an illustration. In the sparse-coding-based methods, let us consider that an f1 × f1 low-resolution patch is extracted from the input image. Then the sparse coding solver, like Feature-Sign, will ﬁrst project the patch onto a (low resolution) dictionary. If the dictionary size is n1, this is equivalent to applying n1 linear ﬁlters (f1 × f1) on the input image (the mean subtraction is also a linear operation so can be absorbed). This is illustrated as the left part of Figure 6.2. The sparse coding solver will then iteratively process the n1 coefﬁcients. The outputs of this solver are n2 coefﬁcients, and usually n2 = n1 in the case of sparse coding. These n2 coefﬁcients are the representation of the high-resolution patch. In this sense, the sparse coding solver behaves as a special case of a non-linear mapping operator, whose spatial support is 1×1. See the middle part of Figure 6.2. However, the sparse coding solver is not feed-forward, i.e., it is an iterative algorithm. On the contrary, our non-linear operator is fully feed-forward and can be computed efﬁciently. If we set f2 = 1, then our non-linear operator can be considered as a pixel -wise fully-connected layer. It is worth noting that “the sparse coding solver” in SRCNN refers to the ﬁrst two layers, but not just the second layer or the activation function (ReLU). Thus, the nonlinear operation in SRCNN is also well optimized through the learning process. The above n2 coefﬁcients (after sparse coding) are then projected onto another (high-resolution) dictionary to produce a high-resolution patch. The overlapping high-resolution patches are then averaged. As discussed above, this is equivalent to linear convolutions on the n2 feature maps. If the high-resolution patches used for reconstruction are of size f3 ×f3, then the linear ﬁlters have an equivalent spatial support of size f3 × f3. See the right part of Figure 6.2. The above discussion shows that the sparse-coding based SR method can be viewed as a kind of convolutional neural network (with a different non-linear mapping). But not all operations have been considered in the optimization in the sparse-coding-based SR methods. On the contrary, in our convolutional neural network, the low-resolution dictionary, high-resolution dictionary, non-linear mapping, together with mean subtraction and averaging, are all involved in the ﬁlters to be optimized. So, our method optimizes an end-to-end mapping that consists of all operations. The above analogy can also help us to design hyperparameters. For example, we can set the ﬁlter size of the last layer to be smaller than that of the ﬁrst layer, and thus we rely more on the central part of the high resolution patch (to the extreme, if f3 = 1, we are using the center pixel with no averaging). We can also set n2 < n1 because it is expected to be sparser. A typical and basic setting is f1 = 9, f2 = 1, f3 = 5, n1 = 64, and n2 = 32 (we evaluate more settings in the experiment section). On the whole, the estimation of a high-resolution pixel utilizes the information of (9 + 5 − 1)2 = 169 pixels. Clearly, the information exploited for reconstruction is comparatively larger than that used in existing external example-based approaches, e.g., using (5+5−1)2 = 81 pixels5. This is one of there reasons why the SRCNN gives superior performance.



responses

of patch

of

neighbouring

patches

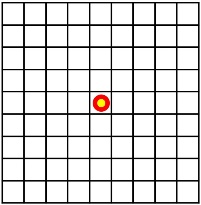


Fig.6.2. Patch extraction Non-linear Reconstruction and representation mapping

**6.3 CONVOLUTIONAL NEURAL NETWORK (CONVNET/CNN)**

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex.

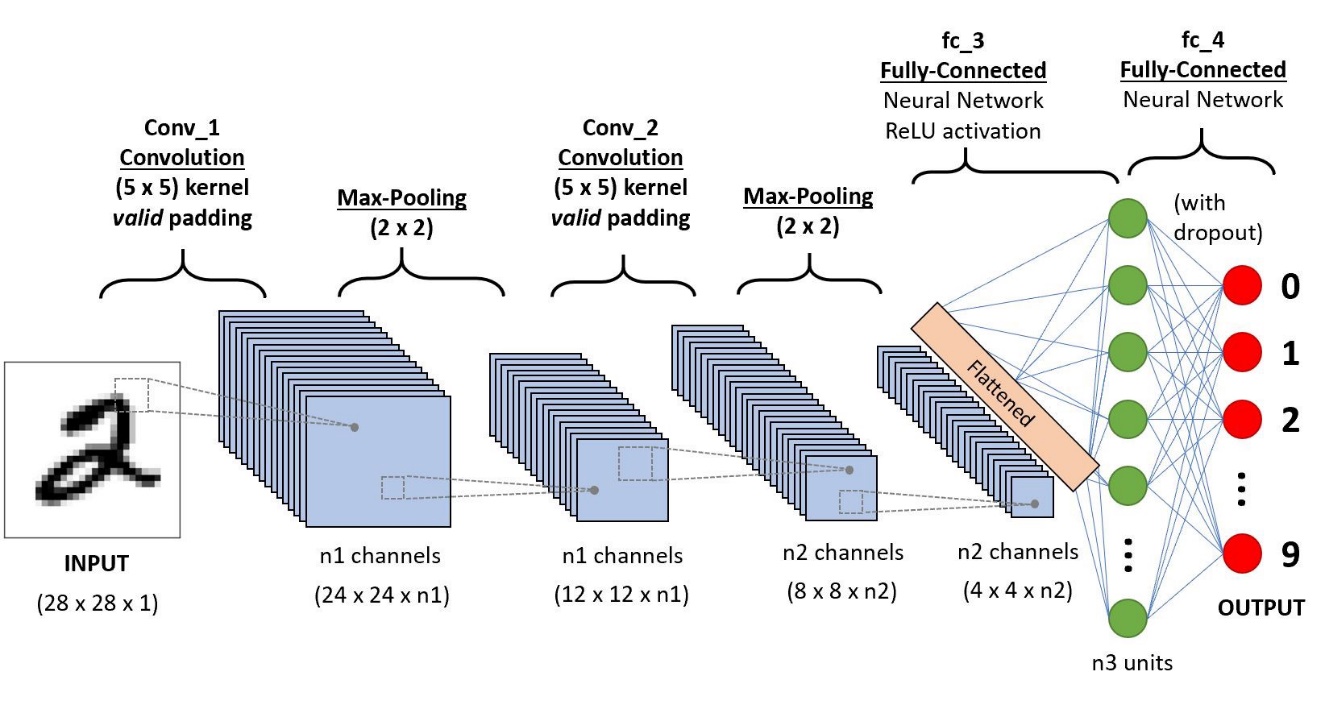


Fig.6.3. Example CNN sequence to classify handwritten digits

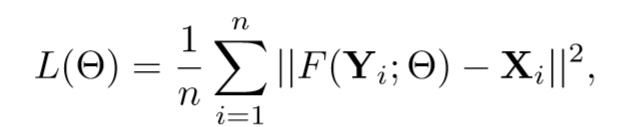
A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, colour, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network, which has the wholesome understanding of images in the dataset, similar to how we would.

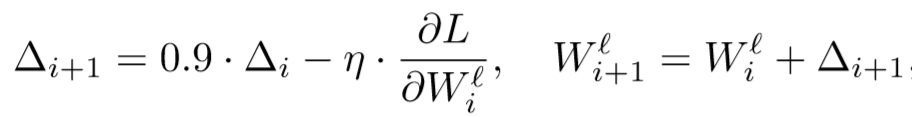
There are two types of results to the operation — one in which the convolved feature is reduced in dimensionality as compared to the input, and the other in which the dimensionality is either increased or remains the same. This is done by applying Valid Padding in case of the former, or Same Padding in the case of the latter.

**6.4 TRAINING**

Learning the end-to-end mapping function F requires the estimation of network parameters Θ = {W1,W2,W3,B1,B2,B3}. This is achieved through minimizing the loss between the reconstructed images F(Y;Θ) and the corresponding ground truth high resolution images X. Given a set of high-resolution images {Xi} and their corresponding low-resolution images {Yi}, we use Mean Squared Error (MSE) as the loss function:



where n is the number of training samples. Using MSE as the loss function favours a high PSNR. The PSNR is a widely-used metric for quantitatively evaluating image restoration quality, and is at least partially related to the perceptual quality. It is worth noticing that the convolutional neural networks do not preclude the usage of other kinds of loss functions, if only the loss functions are derivable. If a better perceptually motivated metric is given during training, it is ﬂexible for the network to adapt to that metric. On the contrary, such a ﬂexibility is in general difﬁcult to achieve for traditional “handcrafted” methods. Despite that the proposed model is trained favouring a high PSNR, we still observe satisfactory performance when the model is evaluated using alternative evaluation metrics, e.g., SSIM, MSSIM. The loss is minimized using stochastic gradient descent with the standard backpropagation. In particular, the weight matrices are updated as



where ` ∈{1,2,3} and i are the indices of layers and iterations, η is the learning rate, and ∂L ∂W` i is the derivative. The ﬁlter weights of each layer are initialized by drawing randomly from a Gaussian distribution with zero mean and standard deviation 0.001 (and 0 for biases). The learning rate is 10−4 for the ﬁrst two layers, and 10−5 for the last layer. We empirically ﬁnd that a smaller learning rate in the last layer is important for the network to converge. In the training phase, the ground truth images {Xi} are prepared as fsub×fsub×c-pixel sub-images randomly cropped from the training images. By “sub-images” we mean these samples are treated as small “images” rather than “patches”, in the sense that “patches” are overlapping and require some averaging as post-processing but “sub-images” need not. To synthesize the low-resolution samples{Yi},weblurasub-image by a Gaussiankernel, sub-sample it by the upscaling factor, and upscale it by the same factor via bicubic interpolation. To avoid border effects during training, all the convolutional layers have no padding, and the network produces a smaller output ((fsub−f1−f2−f3 +3)2×c). The MSE loss function is evaluated only by the difference between the central pixels of Xi and the network output. Although we use a ﬁxed image size in training, the convolutional neural network can be applied on images of arbitrary sizes during testing. We implement our model using the cuda-convnet package. We have also tried the Caffe package and observed similar performance.

**6.5 TRAINING DATA**

As shown in the literature, deep learning generally beneﬁts from big data training. For comparison, we use a relatively small training set that consists of 91 images, and a large training set that consists of 395,909 images from the ILSVRC 2013 ImageNet detection training partition. The size of training sub-images is fsub = 33. Thus the 91-image dataset can be decomposed into 24,800 sub-images, which are extracted from original images with a stride of 14.

Whereas the Image Net provides over 5 million sub-images even using a stride of 33. We use the basic network settings, i.e., f1 = 9, f2 = 1, f3 = 5, n1 = 64, and n2 = 32. The upscaling factor is 3. We use the sparse-coding-based method as our baseline, which achieves an average PSNR value of 31.42 dB. The test convergence curves of using different training sets are shown in Figure 4. The training time on ImageNet is about the same as on the 91-image dataset since the number of back propagations is the same. As can be observed, with the same number of backpropagations (i.e.,8×108), the SRCNN+ImageNet achieves 32.52 dB, higher than 32.39 dB yielded by that trained on 91 images. The results positively indicate that SRCNN performance may be further boosted using a larger training set, but the effect of big data is not as impressive as that shown in high-level vision problems. This is mainly because that the 91 images have already captured sufﬁcient variability of natural images. On the other hand, our SRCNN is a relatively small network (8,032parameters), which could not over ﬁt the 91images (24,800 samples). Nevertheless, we adopt the ImageNet, which contains more diverse data, as the default training set in the following experiments.

**6.6 NUMBER OF LAYERS**

Recent study suggests that CNN could beneﬁt from increasing the depth of network moderately. Here, we try deeper structures by adding another non-linear mapping layer, which has n22 = 16 ﬁlters with size f22 = 1. We conduct three controlled experiments, i.e., 9-1-1-5, 9-3-1-5, 9-5-1-5, which add an additional layer on 9-1-5, 9-3-5, and 9-5-5, respectively. The initialization scheme and learning rate of the additional layer are the same as the second layer. From Figures 6.6.1, 6.6.2 and 6.6.3, we can observe that the four-layer networks converge slower than the three-layer network. Nevertheless, given enough training time, the deeper networks will ﬁnally catch up and converge to the three-layer ones. The effectiveness of deeper structures for super resolution is found not as apparent as that shown in image classiﬁcation. Furthermore, we ﬁnd that deeper networks do not always result in better performance. Speciﬁcally, if we add an additional layer with n22 = 32 ﬁlters on 9-1-5 network, then the performance degrades and fails to surpass the three-layer network. If we go deeper by adding two non-linear mapping layers with n22 = 32 and n23 = 16 ﬁlters on 9-1-5, then we have to set a smaller learning rate to ensure convergence, but we still do not observe superior performance after a week of training. We also tried to enlarge the ﬁlter size of the additional layer to f22 = 3, and explore two deep structures – 9-33-5 and 9-3-3-3.

2

4

6

8

10

12

x(10

8

31

31.5

32

32.5

Number(of(backprops

Average(test(PSNR((dB)

SRCNN((9

−

1

−

5)

SRCNN((9

−

1

−

1

−

5)

SC((31.42(dB)

Fig.6.4. One-layer CNN

1

2

3

4

5

6

7

8

9

10

xS10

8

31.5

32

32.5

NumberSofSbackprops

AverageStestSPSNRS(dB)

SRCNNS(9

−

3

−

5)

SRCNNS(9

−

3

−

1

−

5)

SCS(31.42SdB)

Fig. 6.5. Three Layer CNN

1

2

3

4

5

6

7

8

xR10

8

31.5

32

32.5

NumberRofRbackprops

AverageRtestRPSNRR(dB)

SRCNNR(9

−

5

−

5)

SRCNNR(9

−

5

−

1

−

5)

SCR(31.42RdB)

Fig. 6.6. Five Layer CNN

**CHAPTER 7**

**PACKAGES**

**7.1 TENSORFLOW**

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. ‍ It is a standard expectation in the industry to have experience in TensorFlow to work in machine learning. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

Starting in 2011, Google Brain built DistBelief as a proprietary machine learning system based on deep learning neural networks. Its use grew rapidly across diverse Alphabet companies in both research and commercial applications. Google assigned multiple computer scientists, including Jeff Dean, to simplify and refactor the codebase of DistBelief into a faster, more robust application-grade library, which became TensorFlow. In 2009, the team, led by Geoffrey Hinton, had implemented generalized backpropagation and other improvements which allowed generation of neural networks with substantially higher accuracy, for instance a 25% reduction in errors in speech recognition.

TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017. While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors. During the Google I/O Conference in June 2016, Jeff Dean stated that 1,500 repositories on GitHub mentioned TensorFlow, of which only 5 were from Google.

# 7.2 KERAS

tf.keras is TensorFlow's high-level API for building and training deep learning models. It's used for fast prototyping, state-of-the-art research, and production, with three key advantages:

* **User-Friendly**  
   Keras has a simple, consistent interface optimized for common use cases. It provides clear and actionable feedback for user errors.
* **Modular and composable**

Keras models are made by connecting configurable building blocks together, with few restrictions.

* **Easy to extend**

Write custom building blocks to express new ideas for research. Create new layers, metrics, loss functions, and develop state-of-the-art models.

**7.3 SKIMAGE**

scikit-image is a collection of algorithms for image processing. It is available free of charge and free of restriction. We pride ourselves on high-quality, peer-reviewed code, written by an active community of volunteers.

Scikit-learn is an open source image processing library for python programming language. It includes algorithms for [segmentation](https://en.wikipedia.org/wiki/Image_segmentation), geometric transformations, colour space manipulation, analysis, filtering, morphology, [feature detection](https://en.wikipedia.org/wiki/Feature_detection_(computer_vision)), and more.  It is designed to interoperate with the Python numerical and scientific libraries [NumPy](https://en.wikipedia.org/wiki/NumPy) and [SciPy](https://en.wikipedia.org/wiki/SciPy).

**7.4 MATPLOTLIB**

Matplotlib is a [plotting](https://en.wikipedia.org/wiki/Plotter) [library](https://en.wikipedia.org/wiki/Library_(computer_science)) for the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) programming language and its numerical mathematics extension [NumPy](https://en.wikipedia.org/wiki/NumPy). It provides an [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) [API](https://en.wikipedia.org/wiki/API) for embedding plots into applications like [Tkinter](https://en.wikipedia.org/wiki/Tkinter" \o "Tkinter), [wxPython](https://en.wikipedia.org/wiki/WxPython" \o "WxPython), [Qt](https://en.wikipedia.org/wiki/Qt_(software)), or [GTK+](https://en.wikipedia.org/wiki/GTK%2B). There is also a [procedural](https://en.wikipedia.org/wiki/Procedural_programming) "pylab" interface based on a [state machine](https://en.wikipedia.org/wiki/State_machine) (like [OpenGL](https://en.wikipedia.org/wiki/OpenGL)), designed to closely resemble that of [MATLAB](https://en.wikipedia.org/wiki/MATLAB), though its use is discouraged. [SciPy](https://en.wikipedia.org/wiki/SciPy) makes use of Matplotlib.

Matplotlib was originally written by [John D. Hunter](https://en.wikipedia.org/wiki/John_D._Hunter), has an active development community, and is distributed under a [BSD-style license](https://en.wikipedia.org/wiki/BSD_licenses). Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012, and further joined by Thomas Caswell.

Matplotlib 2.0.x supports Python versions 2.7 through 3.6. Python 3 support started with Matplotlib 1.2. Matplotlib 1.4 is the last version to support Python 2.6. Matplotlib has pledged to not support Python 2 past 2020 by signing the Python 3 Statement.

**7.5 OPENCV**

OpenCV (Open Source Computer Vision Library) is a [library of programming functions](https://en.wikipedia.org/wiki/Library_(computing)) mainly aimed at real-time [computer vision](https://en.wikipedia.org/wiki/Computer_vision). Originally developed by [Intel](https://en.wikipedia.org/wiki/Intel_Corporation), it was later supported by [Willow Garage](https://en.wikipedia.org/wiki/Willow_Garage) then Itseez (which was later acquired by Intel). The library is [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) and free for use under the [open-source](https://en.wikipedia.org/wiki/Open-source_software) [BSD license](https://en.wikipedia.org/wiki/BSD_license).

OpenCV supports some models from deep learning frameworks like TensorFlow, Torch, PyTorch (after converting to an ONNX model) and Caffe according to a defined list of supported layers. It promotes Open Vision Capsules, which is a portable format, compatible with all other formats.

**7.6 NUMPY**

NumPy  is a library for the [Python programming language](https://en.wikipedia.org/wiki/Python_(programming_language)), adding support for large, multi-dimensional [arrays](https://en.wikipedia.org/wiki/Array_data_structure) and [matrices](https://en.wikipedia.org/wiki/Matrix_(math)), along with a large collection of [high-level](https://en.wikipedia.org/wiki/High-level_programming_language) [mathematical](https://en.wikipedia.org/wiki/Mathematics) [functions](https://en.wikipedia.org/wiki/Function_(mathematics)) to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by [Jim Hugunin](https://en.wikipedia.org/wiki/Jim_Hugunin) with contributions from several other developers. In 2005, [Travis Oliphant](https://en.wikipedia.org/wiki/Travis_Oliphant) created NumPy by incorporating features of the competing Num array into Numeric, with extensive modifications. NumPy is [open-source software](https://en.wikipedia.org/wiki/Open-source_software) and has many contributors.

**7.7 OS MODULE**

The OS module in python provides functions for interacting with the operating system. OS, comes under Python’s standard utility modules. This module provides a portable way of using operating system dependent functionality. The \*os\* and \*os.path\* modules include many functions to interact with the file system.

Adam is an adaptive learning rate optimization algorithm that’s been designed specifically for training deep neural networks. First published in 2014, Adam was presented at a very prestigious conference for deep learning practitioners —[ICLR 2015](https://www.iclr.cc/archive/www/doku.php%3Fid=iclr2015:main.html). The paper contained some very promising diagrams, showing huge performance gains in terms of speed of training. However, after a while people started noticing, that in some cases Adam actually finds worse solution than [stochastic gradient descent](https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d). A lot of research has been done to address the problems of Adam.

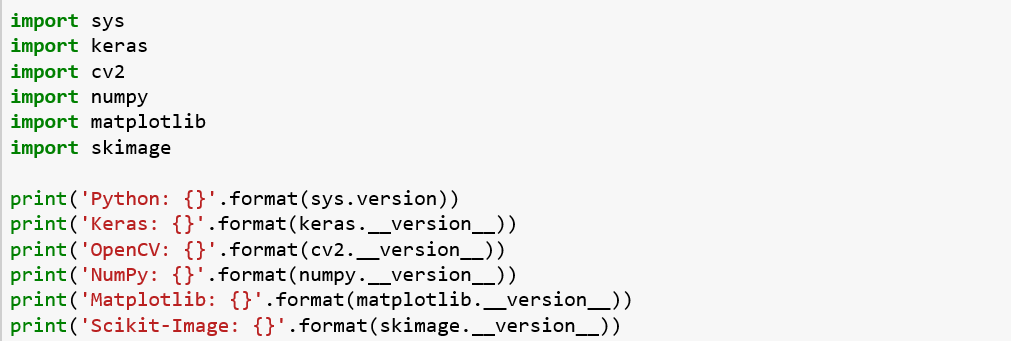
The algorithms leverage the power of adaptive learning rates methods to find individual learning rates for each parameter. It also has advantages of Adagrad , which works really well in settings with sparse gradients, but struggles in non-convex optimization of neural networks, and RMSprop, which tackles to resolve some of the problems of Adagrad and works really well in on-line settings.

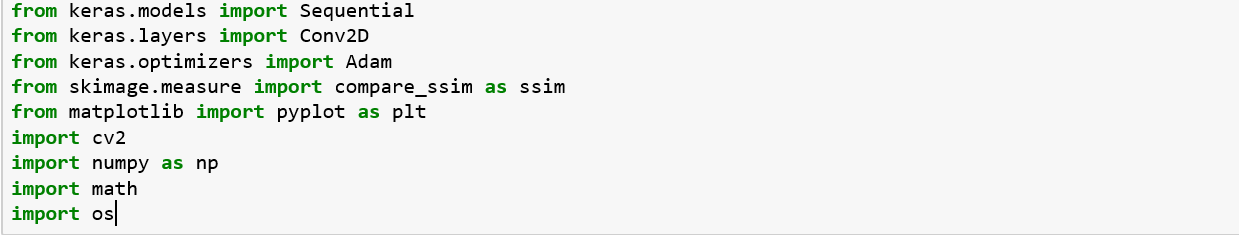
**CHAPTER 8**

**SOURCE CODE IMPLEMENTATION**

# 8.1 IMPORTING PACKAGES

In this first cell, we will import the libraries and packages we will be using in this project and print their version numbers. This is an important step to make sure we are all on the same page; furthermore, it will help others reproduce the results we obtain.

****

****

# 8.2 IMAGE QUALITY METRICS

Define a couple of functions that we can use to calculate the PSNR, MSE, and SSIM. The structural similarity (SSIM) index was imported directly from the scikit-image library; however, we will have to define our own functions for the PSNR and MSE. Furthermore, we will wrap all three of these metrics into a single function that we can call later.

**8.2.1 PEAK SIGNAL-TO-NOISE RATIO (PSNR)**

The term peak signal-to-noise ratio (PSNR) is an expression for the ratio between the maximum possible value (power) of a signal and the power of distorting noise that affects the quality of its representation.  Because many signals have a very wide dynamic range, (ratio between the largest and smallest possible values of a changeable quantity) the PSNR is usually expressed in terms of the logarithmic decibel scale.

2.1. MSE (Mean Square Error)

MSE is the most common estimator of image quality measurement metric. It is a

full reference metric and the values closer to zero are the better.

It is the second moment of the error. The variance of the estimator and its bias

are both incorporated with mean squared error. The MSE is the variance of the

estimator in case of unbiased estimator. It has the same units of measurement as

the square of the quantity being calculated like as variance. The MSE introduces

the Root-Mean-Square Error (RMSE) or Root-Mean-Square Deviation (RMSD)

and often referred to as standard deviation of the variance.

2.1. MSE (Mean Square Error)

MSE is the most common estimator of image quality measurement metric. It is a

full reference metric and the values closer to zero are the better.

It is the second moment of the error. The variance of the estimator and its bias

are both incorporated with mean squared error. The MSE is the variance of the

estimator in case of unbiased estimator. It has the same units of measurement as

the square of the quantity being calculated like as variance. The MSE introduces

the Root-Mean-Square Error (RMSE) or Root-Mean-Square Deviation (RMSD)

and often referred to as standard deviation of the variance.

**8.2.2 MSE (MEAN SQUARE ERROR)**

MSE is the most common estimator of image quality measurement metric. It is a full reference metric and the values closer to zero are the better. It is the second moment of the error. The variance of the estimator and its bias are both incorporated with mean squared error. The MSE is the variance of the estimator in case of unbiased estimator. It has the same units of measurement as the square of the quantity being calculated like as variance. The MSE introduces the Root-Mean-Square Error (RMSE) or Root-Mean-Square Deviation (RMSD) and often referred to as standard deviation of the variance.

Structure Similarity Index Method (SSIM)

Structural Similarity Index Method is a perception based model. In this method,

image degradation is considered as the change of perception in structural infor-

mation. It also collaborates some other important perception based fact such as

luminance masking, contrast masking, etc. The term structural information

emphasizes about the strongly inter-dependant pixels or spatially closed pixels.

These strongly inter-dependant pixels refer some more important information

about the visual objects in image domain. Luminace masking is a term where the

distortion part of an image is less visible in the edges of an image. On the other

hand contrast masking is a term where distortions are also less visible in the

texture of an image. SSIM estimates the perceived quality of images and videos.

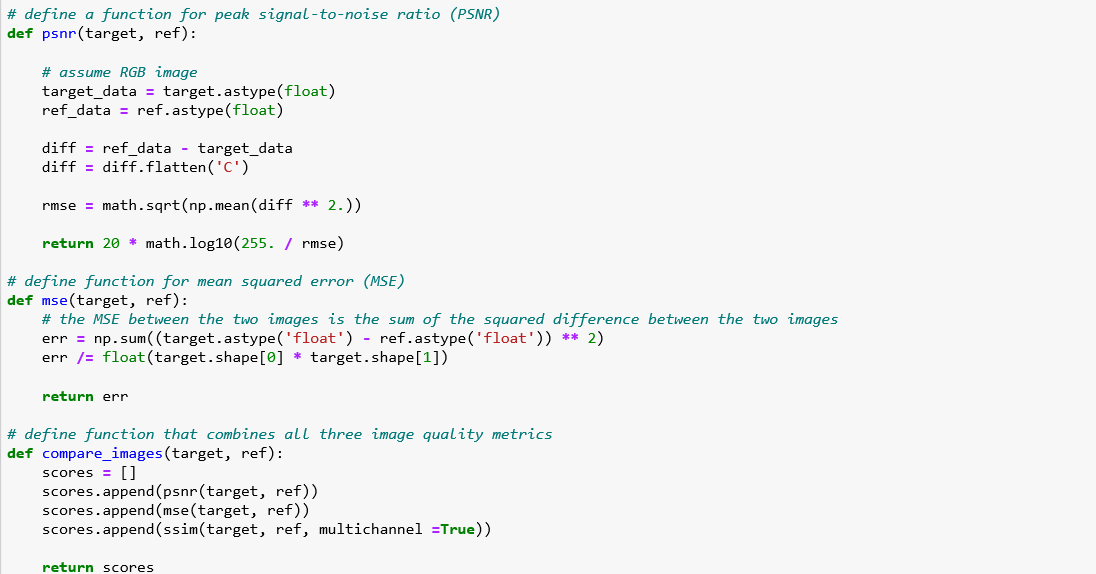
It measures the similarity between two images: the original and the recovered

**8.2.3 STRUCTURE SIMILARITY INDEX METHOD (SSIM)**

Structural Similarity Index Method is a perception based model. In this method, image degradation is considered as the change of perception in structural information. It also collaborates some other important perception based fact such as luminance masking, contrast masking, etc. The term structural information emphasizes about the strongly inter-dependent pixels or spatially closed pixels. These strongly inter-dependent pixels refer some more important information about the visual objects in image domain. Luminance masking is a term where the distortion part of an image is less visible in the edges of an image. On the other hand, contrast masking is a term where distortions are also less visible in the texture of an image. SSIM estimates the perceived quality of images and videos. It measures the similarity between two images: the original and the recovered.

**8.2.4 ROOT MEAN SQUARE ERROR (RMSE)**

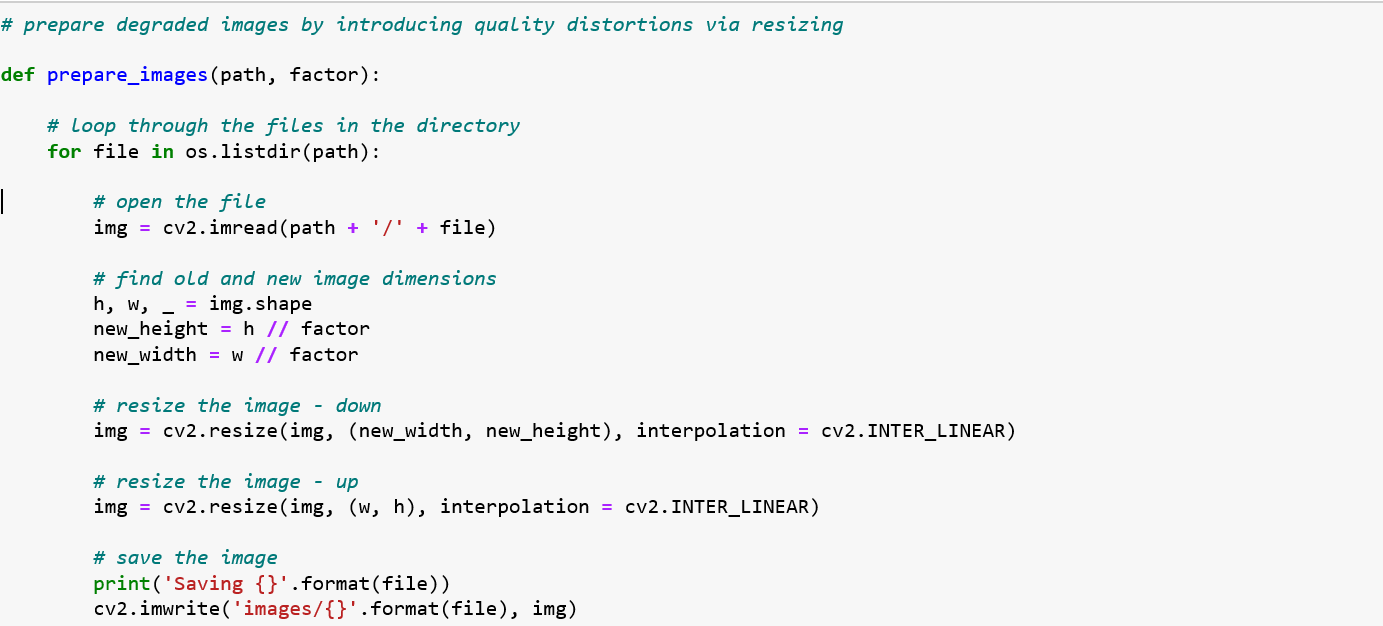
RMSE is the standard deviation of the [residuals](https://www.statisticshowto.com/residual/) ([prediction errors](https://www.statisticshowto.com/prediction-error-definition/)). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the [line of best fit](https://www.statisticshowto.com/line-of-best-fit/). Root mean square error is commonly used in climatology, forecasting, and [regression analysis](https://www.statisticshowto.com/probability-and-statistics/regression-analysis/) to verify experimental results.



# 8.3 PREPARING IMAGES

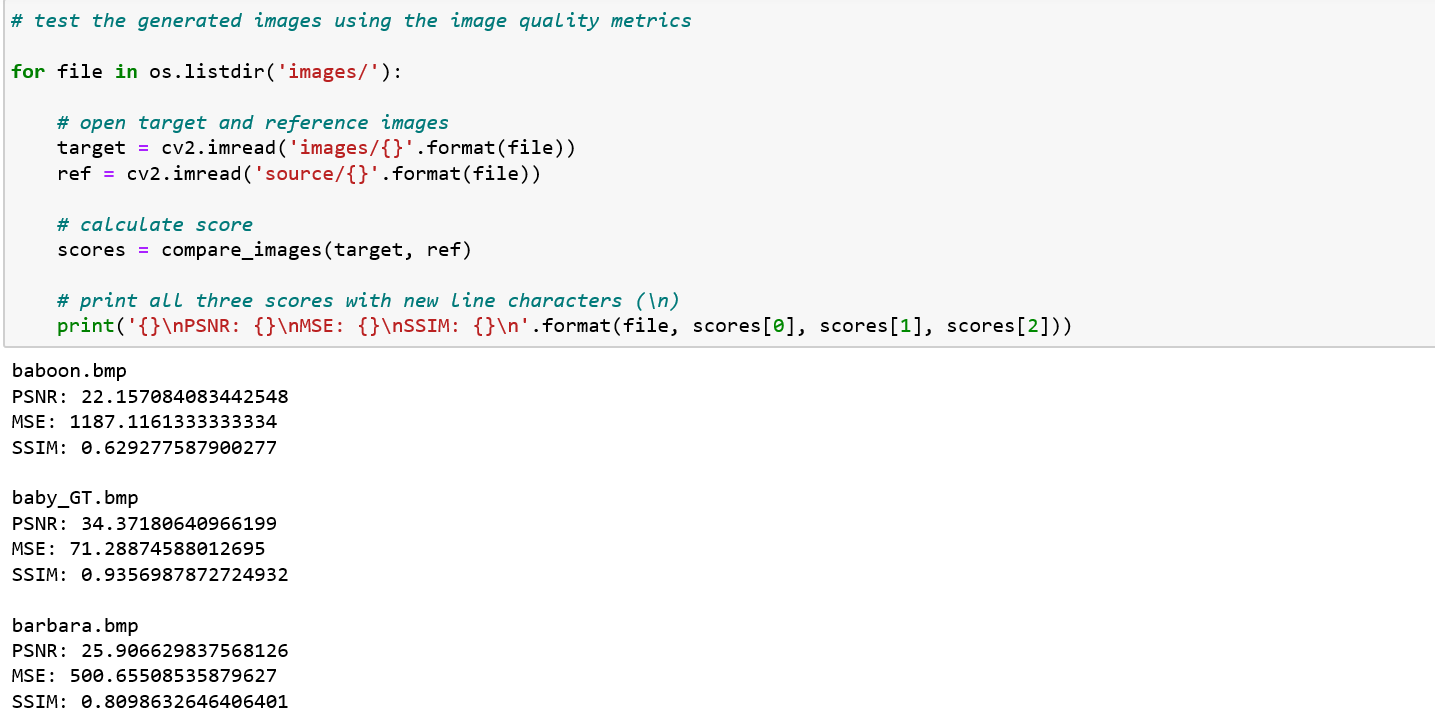
Now that we have some images, we want to produce low-resolution versions of these same images. We can accomplish this by resizing the images, both downwards and upwards, using OpenCV. There are several interpolation methods that can be used to resize images; however, we will be using bilinear interpolation.

Once we produce these low-resolution images, we can save them in a new folder.



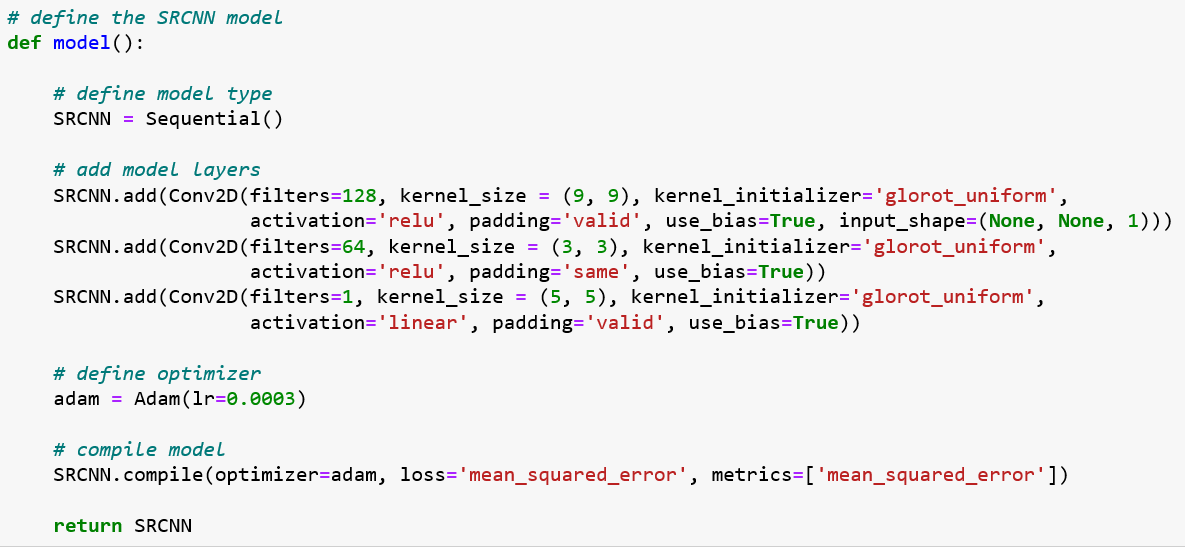
# 8.4 TESTING LOW-RESOLUTION IMAGES

To ensure that our image quality metrics are being calculated correctly and that the images were effectively degraded, let's calculate the PSNR, MSE, and SSIM between our reference images and the degraded images that we just prepared.



**8.5 BUILDING THE SRCNN MODEL**

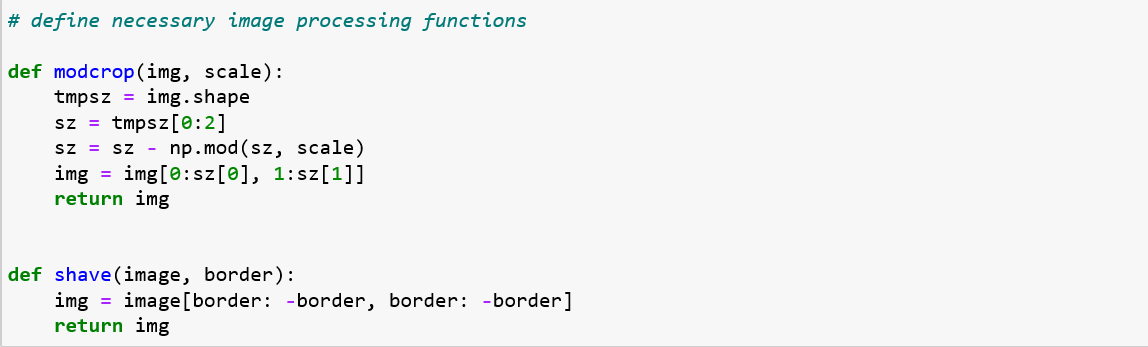
Now that we have our low-resolution images and all three image quality metrics functioning properly, we can start building the SRCNN. In Keras, it’s as simple as adding layers one after the other. The architecture and hyperparameters of the SRCNN network can be obtained from the publication referenced above.

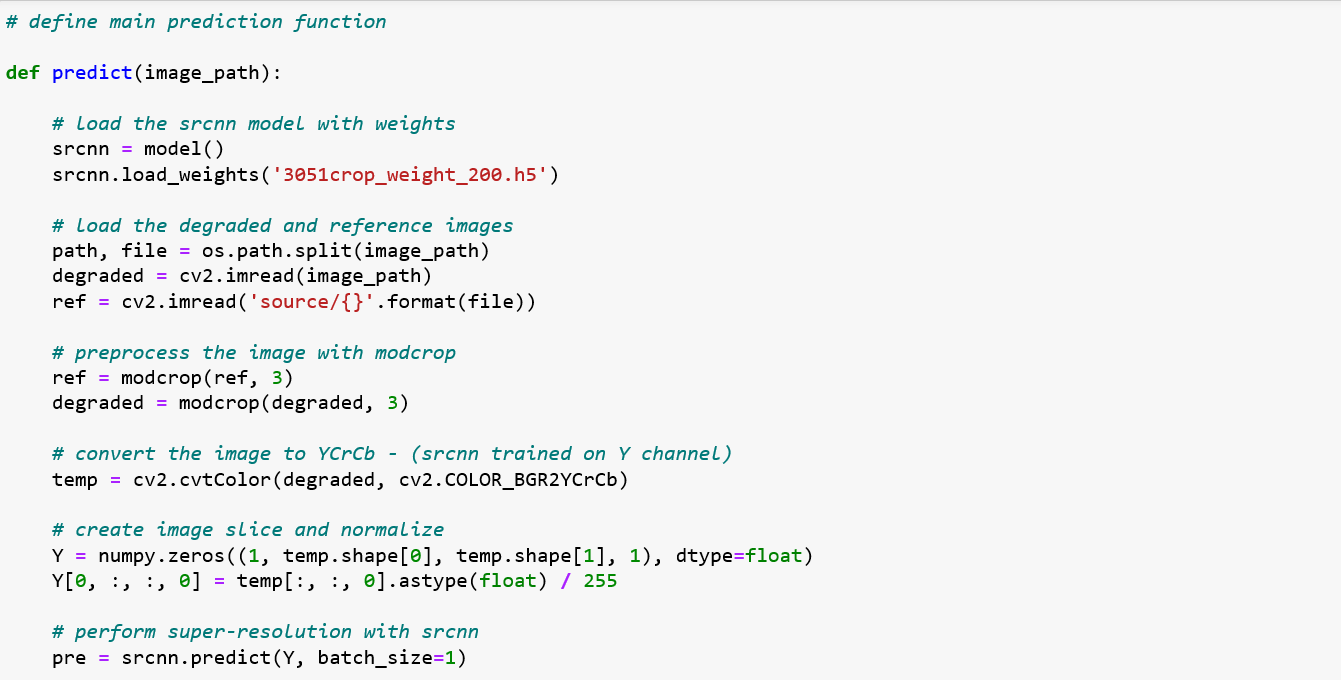


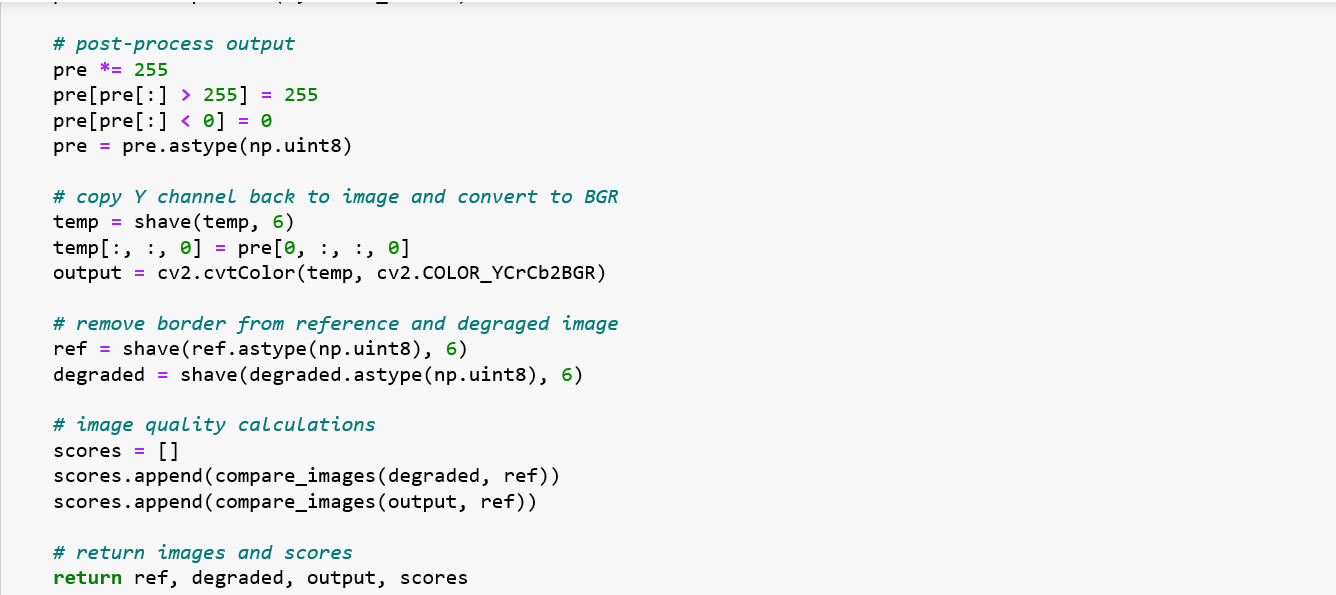
# 8.6 DEPLOYING THE SRCNN

Now that we have defined our model, we can use it for single-image super-resolution. However, before we do this, we will need to define a couple of image processing functions. Furthermore, it will be necessary to pre-process the images extensively before using them as inputs to the network. This processing will include cropping and color space conversions.

Additionally, to save us the time it takes to train a deep neural network, we will be loading pre-trained weights for the SRCNN. Once we have tested our network, we can perform single-image super-resolution on all of our input images. Furthermore, after processing, we can calculate the PSNR, MSE, and SSIM on the images that we produce. We can save these images directly or create subplots to conveniently display the original, low-resolution, and high-resolution images side by side.











**CHAPTER 9**

**UML DIAGRAMS**

**9.1 USE CASE DIAGRAM**

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different [use cases](https://en.wikipedia.org/wiki/Use_case) in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

The purpose of the use case diagrams is simply to provide the high level view of the system and convey the requirements in laypeople's terms for the [stakeholders](https://en.wikipedia.org/wiki/Project_stakeholder). Additional diagrams and documentation can be used to provide a complete functional and technical view of the system.

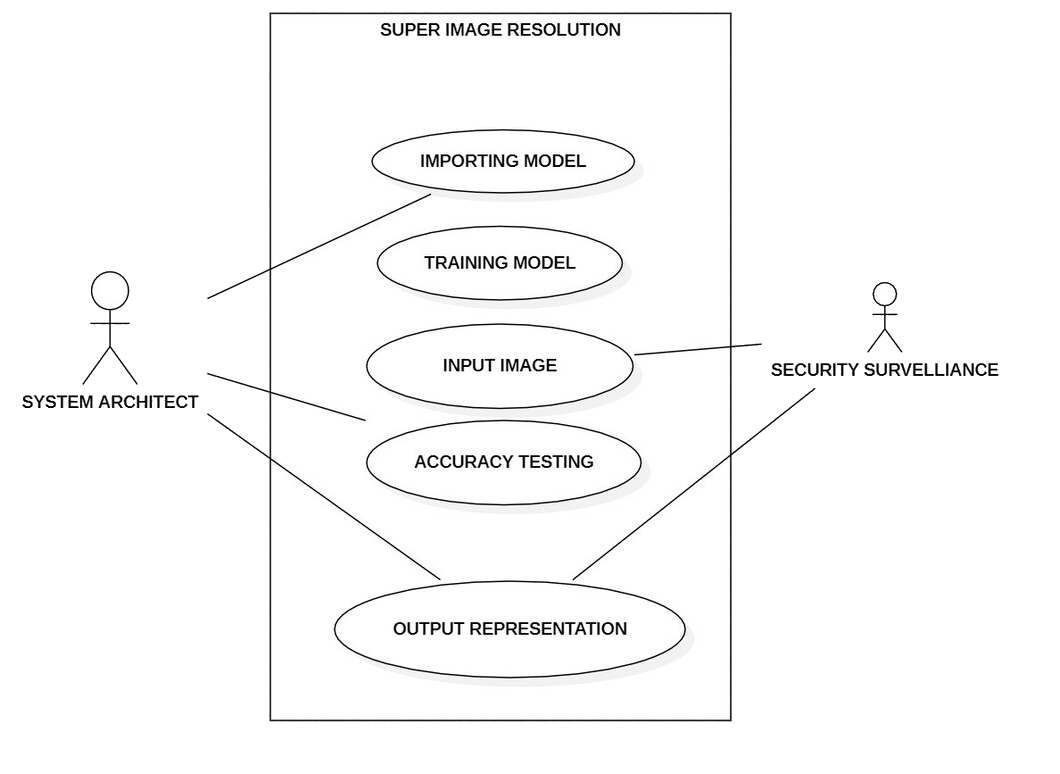


Fig.9.1. use case diagram

**9.2 COMPONENT DIAGRAM**

A component diagram depicts how [components](https://en.wikipedia.org/wiki/Component_(UML)) are wired together to form larger components or [software systems](https://en.wikipedia.org/wiki/Software_system). They are used to illustrate the structure of arbitrarily complex systems.

A component diagram allows verification that a system's required functionality is acceptable. These diagrams are also used as a communication tool between the developer and stakeholders of the system. Programmers and developers use the diagrams to formalize a roadmap for the implementation, allowing for better decision-making about task assignment or needed skill improvements. System administrators can use component diagrams to plan ahead, using the view of the logical software components and their relationships on the system.

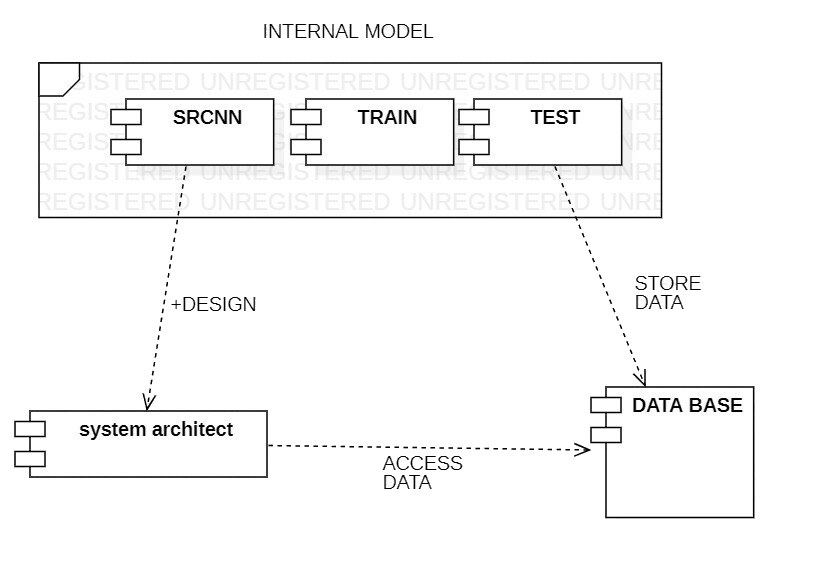


Fig. 9.2. component diagram

**9.3 SEQUENCE DIAGRAM**

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

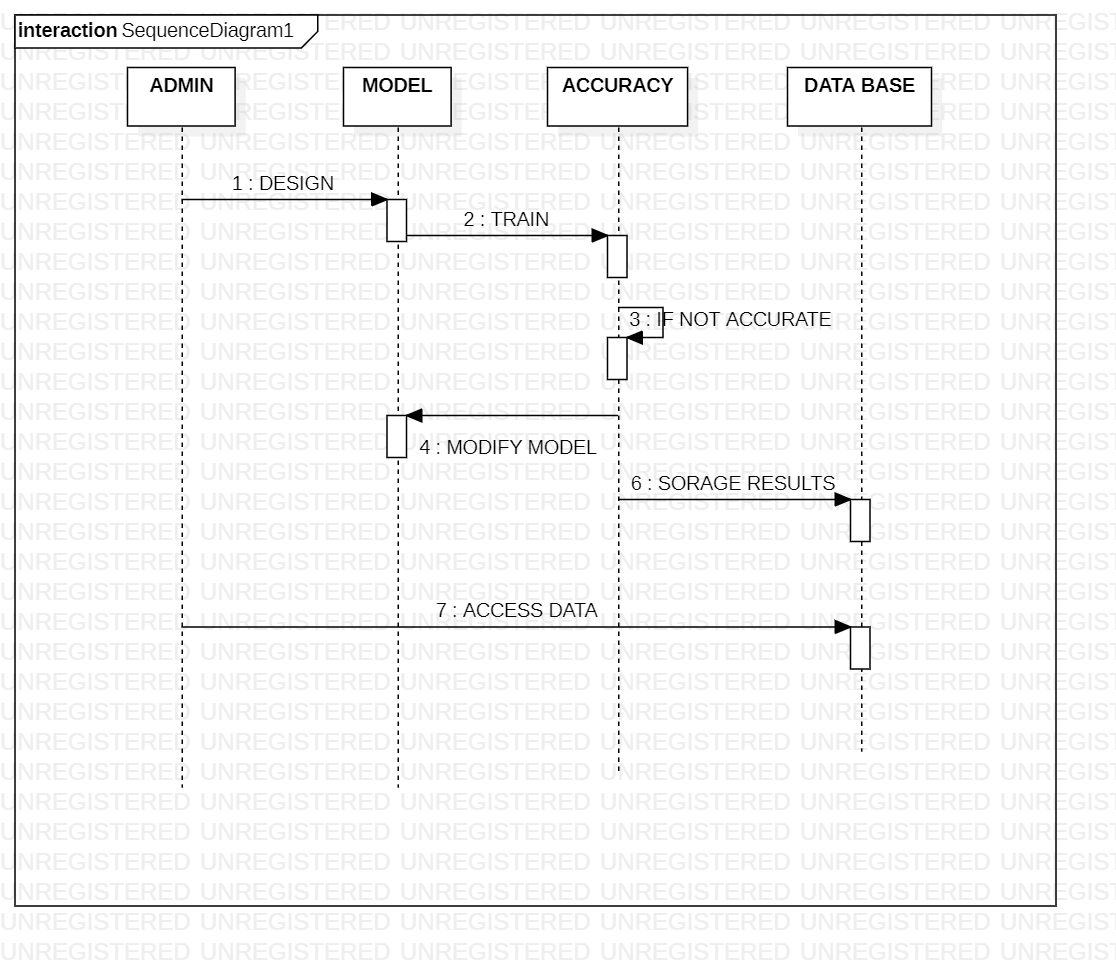


Fig.9.3. sequence diagram

# CHAPTER 10

# SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an un acceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

## **10.1 TYPES OF TESTS**

**10.1.1 UNIT TESTING**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that programming puts produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

### **10.1.2 INTEGRATION TESTING**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

### **10.1.3 FUNCTIONAL TEST**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

* Valid Input: identified classes of valid input must be accepted.
* Invalid Input: identified classes of invalid input must be rejected.
* Functions: identified functions must be exercised.
* Output: identified classes of application outputs must be exercised.
* Systems/Procedure: interfacing systems or procedure must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

### **10.1.4 WHITE BOXTESTING**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or atleast its purpose. It is purpose. It issued to test areas that cannot be reached from a black box level.

### **10.1.5 BLACK BOXTESTING**

### Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**10.2 OTHER TESTING METHODOLOGIES**

### **10.2.1 USER ACCEPTANCE TESTING**

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

**10.2.2 OUTPUT TESTING**

### After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

**CHAPTER 11**

**APPLICATIONS**

**OBJECT DETECTION**

Object detection is a computer vision technique for locating instances of objects in images or videos. Object detection algorithms typically leverage [machine learning](https://in.mathworks.com/discovery/machine-learning.html) or [deep learning](https://in.mathworks.com/discovery/deep-learning.html) to produce meaningful results. When humans look at images or video, we can recognize and locate objects of interest within a matter of moments. The goal of object detection is to replicate this intelligence using a computer.

**SR IN SATELLITE IMAGE PROCESSING**

In the field of satellite imaging it is often desired to have the higher resolution of images. In order to execute this task SR plays very significant role. Satellite image processing area includes the image rectification, restorations, enhancement and also information extraction. All these areas many times need the techniques of super resolution. Super resolved image increases the number of pixels which enhances the display of the digital image i.e. visual interpretation increases. Moreover, It could help in removal of distortions and geographical information can be further enhanced. In the further processing SR can also be combined in further classification of areas or geographical locations. It could also include learning-based techniques which are useful in land map constructions.

**SR IN MEDICAL IMAGE PROCESSING**

In the field of medical industry Super resolution has its own role. It has been that in literature good work has been done to improve the good quality of medical images. As in CT scan, MRI and in other medical imaging techniques high contrast and good enhancement of image is needed this could only be fulfilled by the SR methods.

Most of the images in medical field are of low resolutions, geometric deformations and with low contrast i.e. X ray is having lower contrast, ultrasound having noisy images etc. Moreover, if more time is given for imaging due to patient movement blurring may also be possible. So, in order to get rid of these issues super resolution of images can be

**SR IN MICROSCOPY IMAGE PROCESSING**

Super resolution is also playing an important role in microscopic image processing. In this area recently much advancement has been done as per literature. In order to visualize the biological structures including cell and tissue SR is very useful. Super-resolution fluorescence microscopy one of the very significant field in microscopic imaging. 2014 Nobel Prize in Chemistry also has been given in the area of Super-resolved fluorescence microscopy. In the past time, Florescence microscope is one of the essential tools for examination of the pathways, biological molecules, living cells, tissues, and even whole subjects. It is more useful as compared to electron microscopy. Other techniques like MRI or OCT (optical coherence tomography) can give resolutions in 10s of centimeters and micrometers. However, with super resolutions florescence range can be further increased.

**SR IN MULTIMEDIA INDUSTRY AND VIDEO ENHANCEMENT:**

In today’s time multimedia based applications are increasing day by day. Super resolution is also involving in multimedia industry. In today’s time movies, animations, visual effects all need the HD data. So, SR can also be proved as the useful technique in video enhancements. Many methods used in multimedia-based applications uses the SR method for the enhancement of images and videos. Cell phone-based applications like image or videos also included SR based techniques to enhance their quality.

**SR IN ASTROLOGICAL STUDIES**

In the field of astrological studies, Super resolution is also involved as the significant technique. High resolution astronomical images are always desirable for better computation. Many blurred and noisy images can be combined to get a better view. [19] has used the SR for improvement of quality of astrological images. Many tightly grouped stars and far away objects can be visualizing in better way. In this area many times many unidentified objects could also be visualized in better way.

**OTHER APPLICATIONS**

Beyond the previous applications of SR, It is also having applications in areas like object detection, automotive industry, real time processing, scanning, surveillance, military and in forensics. In the area of surveillance has proposed the method based on Conjugate gradient (CG) optimization. In the similar way in military surveillance also super resolution is used. In automotive industry SR is having its recent applications. In auto classification and robotics, it is acting as the supportive technique. In forensic application also SR based methods are being employed. So overall in many areas SR research-based work is being

In the area of surveillance has proposed the method based on Conjugate gradient (CG) optimization. In the similar way in military surveillance also super resolution is used. In automotive industry SR is having its recent applications. In auto classification and robotics, it is acting as the supportive technique. In forensic application also SR based methods are being employed. So overall in many areas SR research-based work is being involved**.**

**RESOLUTION ENHANCEMENT OF GRAY-LEVEL IMAGES**

To compare the hyper-resolution effect, the down-sampled image was hyper-resolved repeatedly for resizing it to its initial dimensionGray level resolution refers to the predictable or deterministic change in the shades or levels of gray in an image. In short gray level resolution is equal to the number of bits per pixel.

## **CHAPTER 12**

## **PROJECT RESULTS**

**11.1 INPUT IMAGES**

****

****

Fig:11.1 input images

## **11.2 OUTPUT OF SRCNN**







## 

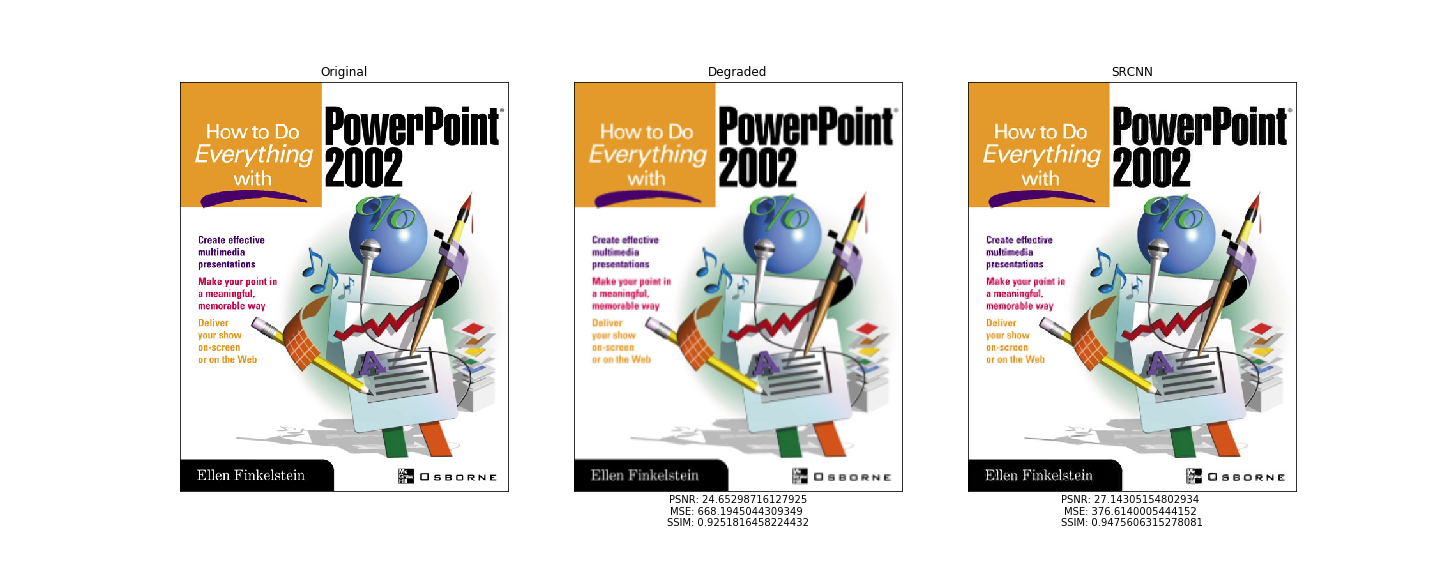


Fig:11.2 output images

## **CHAPTR 13**

## **CONCLUSION AND FUTURE ENHANCEMENTS**

Image super resolution based on deep learning have made breakthroughs in recent years. In this paper, we have given a extensive survey on recent advances in image super resolution with deep learning. We mainly discussed the improvement of supervised super-resolution and unsupervised super resolution, and also introduced some domain specific applications. Despite great success, there are still many unsolved problems. Thus, in this section, we will point out these problems explicitly and introduce some research trends for the future evolution.

We hope that this survey not only provides a better understanding of image super-resolution but also facilitates future research activities and application developments in this field. It divides the recent works into two categories: The deep architectures for simulating the SISR process and the optimization objectives for optimizing the whole process. Despite the promising results reported so far, 14 there are still many underlying problems. We summarize the main challenges into three aspects: the acceleration of deep models, the extensive comprehension of deep models and the criteria for designing and evaluating the objective functions. Along with these challenges, several directions may be further explored in the future.

We have presented a novel deep learning approach for single image super-resolution (SR). We show that conventional sparse-coding-based SR methods can be reformulated into a deep convolutional neural network. The proposed approach, SRCNN, learns an end-to-end mapping between low- and high-resolution images, with little extra pre/post-processing beyond the optimization. With a lightweight structure, the SRCNN has achieved superior performance than the state-of-the-art methods.

We conjecture that additional performance can be further gained by exploring more ﬁlters and different training strategies. Besides, the proposed structure, with its advantages of simplicity and robustness, could be applied to other low-level vision problems, such as image deblurring or simultaneous SR+denoising. One could also investigate a network to cope with different upscaling factors.

**LEARNED FILTERS FOR SUPER-RESOLUTION**

For instance, if we divide the ﬁlters a and f are like Laplacian/Gaussian ﬁlters, the ﬁlters b, c, and d are like edge detectors at diﬀerent directions, and the ﬁlter e is like a texture extractor. We observe some “dead” ﬁlters, whose weights are all nearly zeros. Nevertheless, patterns may emerge in some of these dead ﬁlters given long enough training time. We will investigate this phenomenon in future work.

**CHAPTER 14**

**REFERENCES**

1. Chao Dong, Chen Change Loy, Member, IEEE, Kaiming He, Member, IEEE, and Xiaoou Tang, Fellow, IEEE
2. Isaac J.S and Kulkarni. R, “Super resolution techniques for medical image processing,” in ICTSD, 2015.
3. Huang. Y, Shao.L, and Frangi A.F, “Simultaneous super-resolution and cross-modality synthesis of 3d medical images using weakly-supervised joint convolutional sparse coding,” in CVPR, 2017.
4. Lin. F, Fookes. C, Chandran. V, and Sridharan. S, “Super-resolved faces for improved face recognition from surveillance video,” in ICB, 2007.
5. Zhang. L, Zhang. H, Shen. H, and Li. P, “A super-resolution reconstruction algorithm for surveillance images,” Elsevier Signal Processing, vol. 90, 2010.
6. Rasti. P, Uiboupin. T, Escalera. S, and Anbarjafari. G, “Convolutional neural network super resolution for face recognition in surveillance monitoring,” in AMDO, 2016.
7. Zhang. H, Liu. D, and Xiong. Z, “Convolutional neural network-based video super-resolution for action recognition,” in FG, 2018.
8. M. Haris, G. Shakhnarovich, and N. Ukita, “Task-driven super resolution: Object detection in low-resolution images,”
9. M. S. Sajjadi, B. Schölkopf, and M. Hirsch, “Enhancenet: Single image super-resolution through automated texture synthesis,” in ICCV, 2017.
10. Y. Zhang, Y. Bai, M. Ding, and B. Ghanem, “Sod-mtgan: Small object detection via multi-task generative adversarial network,” in ECCV, 2018.
11. R. Keys, “Cubic convolution interpolation for digital image processing,” IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 29, 1981.
12. E. Duchon, “Lanczos filtering in one and two dimensions,” Journal of Applied Meteorology, vol. 18, 1979.
13. M. Irani and S. Peleg, “Improving resolution by image registration,” CVGIP: Graphical Models and Image Processing, vol. 53, 1991.