

CONVERTING DEGRADED IMAGES TO HIGH RESOLUTION IMAGES USING CONVOLUTIONAL NEURAL NETWORK

Submitted in partial fulfillment of the requirements
for the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

HARSHA VARDHAN G (Reg. No. 37110270)

JUJJAVARAPU DILEEP KUMAR (Reg. No. 37110298)



**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC JEPPIAAR NAGAR, RAJIV
GANDHI SALAI, CHENNAI – 600 119**

APRIL 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **HARSHA VARDHAN G (Reg. No. 37110270)** and **JUJJAVARAPU DILEEP KUMAR(Reg. No. 37110298)** who carried out the project entitled "**CONVERTING DEGRADED IMAGES TO HIGH RESOLUTION IMAGES USING CONVOLUTIONAL NEURAL NETWORK**" under my supervision from **November 2020** to **April 2021**.

Internal Guide

Dr. A. Siva Sangari, M.E., Ph.D.,

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

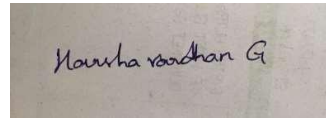
External Examiner

DECLARATION

I **HARSHA VARDHAN G** hereby declare that the Project Report entitled **“CONVERTING DEGRADED IMAGES TO HIGH RESOLUTION IMAGES USING CONVOLUTIONAL NEURAL NETWORK”** is done by me and my partner **JUJJAVARAPU DILEEP KUMAR** under the guidance of **Dr. A. SIVA SANGARI, M.E., Ph.D.,**(Internal Guide) is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

A rectangular box containing a handwritten signature in black ink. The signature appears to read "Harsha vardhan G".

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of Sathyabama Institute of Science and Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean, School of Computing, Dr. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. A. Siva Sangari, M.E., Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

In uncertain times like natural disaster satellite imagery plays a key role in navigating the places which affected the most and also to monitor the fore coming situation, to get better insights from the images, it need be convinible that the quality of the image meets the quality that is required. Presently, we have wide range of approaches to restore the good quality image from distorted image using different Deep learning techniques. To solve the problem with degraded images, we have different methods like Full resolution convolutional neural network, Deep resolution convolution neural network and Very deep resolution neural network and PSNR (Peak signal to noise ratio). Among all PSNR provide better colors. But PSNR absences in providing better quality of textures. To overcome this problem, we have proposed a model to restore the quality of an image using SRCNN (Super Resolution Convolution Neural Network). Super resolution is one of the image restoration technique where we will split each layer and then will perform end-to-end mapping in each layer to obtain a high-quality image from a diminished one. With this proposed system we will restore the better texture for an image with input as degraded image and yields the similar scope image as output with High tenacity.

TABLE OF CONTENTS

ABSTRACT	v
LIST OF FIGURES	viii

CHAPTER No.	TITLE	PAGE No.
1.	INTERODUCTION	1
	1.1 MACHINE LEARNING	1
	1.2 DEEP LEARINIG	2
	1.3 CONVOLUTION NEURAL NETWORK	3
	1.4 SUPER RESOLUTION CNN	6
2.	AIM AND SCOPE	7
	2.1 AIM OF PROJECT	7
	2.1.1 OBJECTIVE	7
	2.2 SCOPE	7
	2.2.1 ADVANTAGES	8
	2.3 BENEFITTING DISASTER MANAGEMENT AND IMAGE PROCESSING SOFTWARE'S	8
3.	SYSTEM DESIGN AND METHODOLOGY	9
	3.1 EXISTING SYSTEM	9
	3.1.1 CONVOLUTIONAL LAYERS	9
	3.1.2 POOLING LAYERS	10
	3.1.3 FULLY CONNECTED LAYERS	10
	3.1.4 RECEPTIVE FIELD	10
	3.1.5 WEIGHTS	11
	3.2.1 VDSR (VERY DEEP SUPER RESOLUTION)	11
	3.2.2 FRCNN (FASTER R-CNN)	12
	3.2.3 PSNR (PEAK SIGNAL-TO-NOISE-RATIO)	15
	3.2.4 SRCNN (SUPER RESOLUTION CONVOLUTIONAL NEURAL NETWORK)	17
4.	SOFTWARE DEVELOPMENT METHODOLOGY	21
	4.1 DESCRIPTION OF DIAGRAM	21
	4.2 FORMULATION	24

4.2.1	PATCH EXTRACTION AND REPRESENTATION	24
4.2.2	NON-LINEAR MAPPING	25
4.2.3	RECONSTRUCTION	26
4.2.4	RELATIONSHIP TO SPACE-CODING- BASED METHODES	26
4.2.5	TRAINING	28
4.2.6	LEARNED FILTERS FOR SUPER- RESOLUTION	30
4.2.7	MODEL AND PERFORMANCE TRADE- OFFS	31
5.	RESULTS AND DISCUSSION	35
6.	CONCLUSION AND FUTURE WORK	40
	REFERENCES	41
	APPENDIX	
	A .PLAGARISM REPORT	43
	B .PAPER	44
	C .SOURCE CODE	48

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
3.2.1	VDSR NETWORK ARCHITECTURE	12
3.2.2.1	EXAMPLE OF SEARCH SELECTIVE	13
3.2.2.2	FIRST STEP OF FRCNN	14
3.2.2.3	ONE ANCHOR PROJECTED TO THE ORIGINAL IMAGE	14
3.1.6.4	R-CNN ARCHITECTURE	15
3.2.3	PSNR BEHAVIOR WITH RESPECT TO BITRATE	17
4.1	FLOW DIAGRAM	21
4.2.4	SRCNN EXTRACTION	27
4.2.7.1	A LARGE FILTER LEADS TO BETTER RESULTS	31
4.2.7.2	COMPARISONS BETWEEN THREE-LAYER AND FOUR-LAYER NETWORKS	33
5.1	INITIALIZING THE QUALITY METRICS	35
5.2	PSNR VALUES FOR DEGRADED IMAGES	38
5.3	IMAGE COMPARISON BEFORE AND AFTER IMPLEMENTING THE MODEL	39
5.4	IMAGE COMPARISON BEFORE AND AFTER IMPLEMENTING THE MODEL	39

CHAPTER 1

INTRODUCTION

1.1 MACHINE LEARNING

Machine learning (ML) is the study of computer algorithms that improve automatically through experience. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

A subset of machine learning is closely related to computational statistics, which focuses on making predictions using computers; but not all machine learning is statistical learning. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics. Machine learning involves computers discovering how they can perform tasks without being explicitly programmed to do so. It involves computers learning from data provided so that they carry out certain tasks. For simple tasks assigned to computers, it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand; on the computer's part, no learning is needed. For more advanced tasks, it can be challenging for a human to manually create the needed algorithms. In practice, it can turn out to be more effective to help the machine develop its own algorithm, rather than having human programmers specify every needed step. The discipline of machine learning employs various approaches to teach computers to accomplish tasks where no fully

satisfactory algorithm is available. In cases where vast numbers of potential answers exist, one approach is to label some of the correct answers as valid. This can then be used as training data for the computer

to improve the algorithm(s) it uses to determine correct answers. For example, the MNIST dataset of handwritten digits.

Machine learning approaches are traditionally divided into three broad categories, depending on the nature of the "signal" or "feedback" available to the learning system:

Supervised learning: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.

Unsupervised learning: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).

Reinforcement learning: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). As it navigates its problem space, the program is provided feedback that's analogous to rewards, which it tries to maximize.

1.2 DEEP LEARNING

Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised.

Deep-learning architectures such as deep neural networks, deep belief networks, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, machine vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance.

Artificial neural networks (ANNs) were inspired by information processing and distributed communication nodes in biological systems. ANNs have various differences

from biological brains. Specifically, neural networks tend to be static and symbolic, while the biological brain of most living organisms is dynamic (plastic) and analogue.

The adjective "deep" in deep learning refers to the use of multiple layers in the network. Early work showed that a linear perceptron cannot be a universal classifier, and then that a network with a nonpolynomial activation function with one hidden layer of unbounded width can on the other hand so be. Deep learning is a modern variation which is concerned with an unbounded number of layers of bounded size, which permits practical application and optimized implementation, while retaining theoretical universality under mild conditions. In deep learning the layers are also permitted to be heterogeneous and to deviate widely from biologically informed connectionist models, for the sake of efficiency, trainability and understandability, whence the "structured" part. Most modern deep learning models are based on artificial neural networks, specifically convolutional neural networks (CNN)s, although they can also include propositional formulas or latent variables organized layer-wise in deep generative models such as the nodes in deep belief networks and deep Boltzmann machines.

In deep learning, each level learns to transform its input data into a slightly more abstract and composite representation. In an image recognition application, the raw input may be a matrix of pixels; the first representational layer may abstract the pixels and encode edges; the second layer may compose and encode arrangements of edges; the third layer may encode a nose and eyes; and the fourth layer may recognize that the image contains a face. Importantly, a deep learning process can learn which features to optimally place in which level on its own.

1.3 CONVOLUTIONAL NEURAL NETWORK

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are

also known as shift invariant or space invariant artificial neural networks (SIANN), based on the shared-weight architecture of the convolution kernels that scan the hidden layers and translation invariance characteristics. They have applications in image and video

recognition, recommender systems, image classification, Image segmentation, medical image analysis, natural language processing, brain-computer interfaces, and financial time series.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include varying the weights as the loss function gets minimized while randomly trimming connectivity. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble patterns of increasing complexity using smaller and simpler patterns embossed in the filters. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically, this includes a layer that does multiplication or other dot product, and its activation function is commonly ReLU. This is followed by other convolution layers such as pooling layers, fully connected layers and normalization layers.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns to optimize the filters or convolution

kernels that in traditional algorithms are hand-engineered. This independence from prior knowledge and human intervention in feature extraction is a major advantage.

The time delay neural network (TDNN) was introduced in 1987 by Alex Waibel et al. and was the first convolutional network, as it achieved shift invariance. It did so by utilizing weight sharing in combination with Backpropagation training. Thus, while also using a pyramidal structure as in the neocognitron, it performed a global optimization of the weights instead of a local one.

TDNNs are convolutional networks that share weights along the temporal dimension. They allow speech signals to be processed time-invariantly. In 1990 Hampshire and Waibel introduced a variant which performs a two dimensional convolution. Since these TDNNs operated on spectrograms, the resulting phoneme recognition system was invariant to both shifts in time and in frequency. This inspired translation invariance in image processing with CNNs. The tiling of neuron outputs can cover timed stages. TDNNs now achieve the best performance in far distance speech recognition. In 1990 Yamaguchi et al. introduced the concept of max pooling. They did so by combining TDNNs with max pooling in order to realize a speaker independent isolated word recognition system. In their system they used several TDNNs per word, one for each syllable. The results of each TDNN over the input signal were combined using max pooling and the outputs of the pooling layers were then passed on to networks performing the actual word classification. A system to recognize hand-written ZIP Code numbers[35] involved convolutions in which the kernel coefficients had been laboriously hand designed. Yann LeCun et al. (1989) used back-propagation to learn the convolution kernel coefficients directly from images of hand-written numbers. Learning was thus fully automatic, performed better than manual coefficient design, and was suited to a broader range of image recognition problems and image types. This approach became a foundation of modern computer vision.

1.4 SUPER RESOLUTION CONVOLUTIONAL NEURAL NETWORK

Image super-resolution (SR) is to convert a given low-resolution (LR) image into a corresponding high-resolution (HR) image by an algorithm [1, 2, 3]. In practice, SR has a wide range of application scenarios and application prospects, such as video, security and medical imaging where demand more image details. In early, traditional methods of SR, including interpolation, sparse coding and other methods, require rigorous algorithm to extract engineered features. With the development of CNN, using convolutional neural network. SRCNN is the first method to apply CNN to image super-resolution (SR) and achieves the state-of-the-art performance. It fully demonstrates the effectiveness of CNN for SR and is an end-to-end method. End-to-end means inputting a LR image and then outputting a HR image, unnecessary to care about the middle of the conversion process [4,5]. The methods for SR based on CNN, is only needing to give images and the architecture of network for training, which makes these methods have a wider range of applicability. So more and more researchers have begun to use deep learning to do the image studies, not only for SR, and have made a lot of progress. In the field of image super-resolution (SR), the depth of convolutional networks has increased from SRCNN's 3 layers to currently 20-30 layers [7, 8, and 9]. Recently many methods using CNN for SR have been proposed. Among them, VDSR shows the best performance. VDSR is proposed by Kim et al. in 2016 and the depth of its network is 20 layers [9]. VDSR uses cascading the same convolutional layers and learning residuals to reduce the super-parameters of networks and speed-up the training. Although the promotion of VDSR for SR is noticeable, but the amount of the network parameters is still huge. If continuing to increase the depth of the network, we will face the serious growth of computing complexity, which makes the training of the network difficult to converge and even effects the performance. bad. As known, huge growth of the amounts of parameters is accompanied by over-fitting. We would like to proposed a strategy to reduce the amount of the network parameters, while the

performance is as good as before. In the field of classification and recognition, deep learning methods have also take place of the traditional methods. And some methods based on CNN have using topology strategy to improve the architecture of networks.

CHAPTER 2

AIM AND SCOPE

2.1 AIM OF PROJECT

The Aim of this project is to restore the quality of an image by end-to-end mapping between the low/high-resolution images. The mapping is represented as a deep convolutional neural network (CNN) that takes the low-resolution image as the input and outputs the high-resolution one. We further show that traditional sparse-coding-based SR methods can also be viewed as a deep convolutional network. But unlike traditional methods that handle each component separately, our method jointly optimizes all layers. Our deep CNN has a lightweight structure, yet demonstrates state-of-the-art restoration quality, and achieves fast speed for practical on-line usage. We explore different network structures and parameter settings to achieve trade-offs between performance and speed. Moreover, we extend our network to cope with three color channels simultaneously, and show better overall reconstruction quality.

2.1.1 OBJECTIVES

The proposed model “converting degraded images to high resolution images using convolutional neural network” is based on SRCNN where the Low resolution image is taken as input. The image is blurred completely in the next process. This image is given as input for the next three convolution layers, these layers will perform the feature extraction, remove the Negative values by using ReLu (Rectified linear units) and mound the image into required size as same as original image. The process of image modCrop is done by using padding and pooling.

2.2 SCOPE

Images resampled with the bi-cubic interpolation will be having very smoother surface and have very few interpolation artifacts. So, we need to choose a method

Bi-cubic interpolated image as an input and is sent to three convolution layers for further processing.

2.2.1 ADVANTAGES

- Faster than VDSR, FRCNN and PSNR
- Better textures
- Easy to implement
- Huge advantage in many industry Applications

2.3 Benefitting Disaster management and Image Processing Software's

Image restoration is different from image enhancement in that the latter is designed to emphasize features of the image that make the image more pleasing to the observer, but not necessarily to produce realistic data from a scientific point of view. Image enhancement techniques (like contrast stretching or de-blurring by a nearest neighbor procedure) provided by imaging packages use no a priori model of the process that created the image.

With image enhancement noise can effectively be removed by sacrificing some resolution, but this is not acceptable in many applications. In a fluorescence microscope, resolution in the z-direction is bad as it is. More advanced image processing techniques must be applied to recover the object. The objective of image restoration techniques is to reduce noise and recover resolution loss. Image processing techniques are performed either in the image domain or the frequency domain.

Most images and video are transmitted or stored in lossy compressed formats. Some of these compression artifacts (e.g., blocking artifacts in JPEG) are quite visible. Reducing them may be thought of as an image restoration problem. These techniques are also known as “postprocessing of compressed images.” Other types of impairment are due to data transmission. For instance, blocks of image data may be lost due to packet loss in ATM networks over which the image is transmitted. Assuming the receiver knows which packets have been lost, the degradation model

mapping the original x to the received y is again deterministic and many-to-one.

CHAPTER 3

SYSTEM DESIGN & METHODOLOGY

3.1 EXISTING SYSTEM

Image restoration techniques has many approaches to solve the problem with image degradation. But, commonly each model follows the Convolution network to solve the problem. In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically this includes a layer that does multiplication or other dot product, and its activation function is commonly ReLU. This is followed by other convolution layers such as pooling layers, fully connected layers and normalization layers.

3.1.1 CONVOLUTIONAL LAYERS

In a CNN, the input is a tensor with shape (number of images) x (image height) x (image width) x (input channels). After passing through a convolutional layer, the image becomes abstracted to a feature map, with shape (number of images) x (feature map height) x (feature map width) x (feature map channels). A convolutional layer within a neural network should have the following attributes:

- Convolutional filters/kernels defined by a width and height (hyper-parameters).
- The number of input channels and output channels (hyper-parameter).
- The depth of the convolution kernel/filter (the input channels) must equal the number channels (depth) of the input feature map.

The hyperparameters of the convolution operation, like padding size and stride.

Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus. Each convolutional neuron processes data only for its receptive field. Although fully connected feedforward neural networks can be used to learn features and classify data, this architecture is impractical for images. It would require a very high number

of neurons, even in a shallow architecture, due to the very large input sizes associated

with images, where each pixel is a relevant variable. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10,000 weights for each neuron in the second layer. Instead, convolution reduces the number of free parameters, allowing the network to be deeper. For example, regardless of image size, tiling 5 x 5 region, each with the same shared weights, requires only 25 learnable parameters. Using regularized weights over fewer parameters avoids the vanishing gradient and exploding gradient problems seen during backpropagation in traditional neural networks

3.1.2 POOLING LAYERS

Convolutional networks may include local or global pooling layers to streamline the underlying computation. Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, typically 2 x 2. Global pooling acts on all the neurons of the convolutional layer. There are two common types of pooling: max and average. Max pooling uses the maximum value of each cluster of neurons at the prior layer, while average pooling instead uses the average value

3.1.3 FULLY CONNECTED LAYERS

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is the same as a traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

3.1.4 RECEPTIVE FIELD

In neural networks, each neuron receives input from some number of locations in the previous layer. In a fully connected layer, each neuron receives input from every neuron of the previous layer. In a convolutional layer, each neuron receives input from only a restricted area of the previous layer called the neuron's receptive field.

Typically the area is a square (e.g., 5 by 5 neurons). (So, in a fully connected layer, the receptive field is the entire previous layer). Thus, in each convolutional layer, each neuron takes input from a larger area of pixels in the input image than previous layers. This is due to

applying the convolution over and over, which takes into account the value of a pixel and its surrounding pixels. In dilated CNNs, the number of pixels in the receptive field remains invariant but the field gets more sparsely populated as its dimensions grow when assessing the influence of different layers.

3.1.5 WEIGHTS

Each neuron in a neural network computes an output value by applying a specific function to the input values coming from the receptive field in the previous layer. The function that is applied to the input values is determined by a vector of weights and a bias (typically real numbers). Learning consists of iteratively adjusting these biases and weights.

The vector of weights and the bias are called filters and represent particular features of the input (e.g., a particular shape). A distinguishing feature of CNNs is that many neurons can share the same filter. This reduces memory footprint because a single bias and a single vector of weights are used across all receptive fields sharing that filter, as opposed to each receptive field having its own bias and vector weighting.

3.2.1 VDSR (Very Deep Super Resolution)

VDSR is a deep learning approach for enlarging an image. It has 20 weight layers which is much deeper compared with SRCNN which only got 3 layers. Sometimes, we only got a poor image and we want to have digital enlargement (zoom in), but the image gets blurred when zoomed in. This is because the conventional interpolation or enlargement of a small image to become a large image, will get a poor image quality. With VDSR, we can obtain a high-resolution (HR) image with high quality from a low resolution (LR) image.

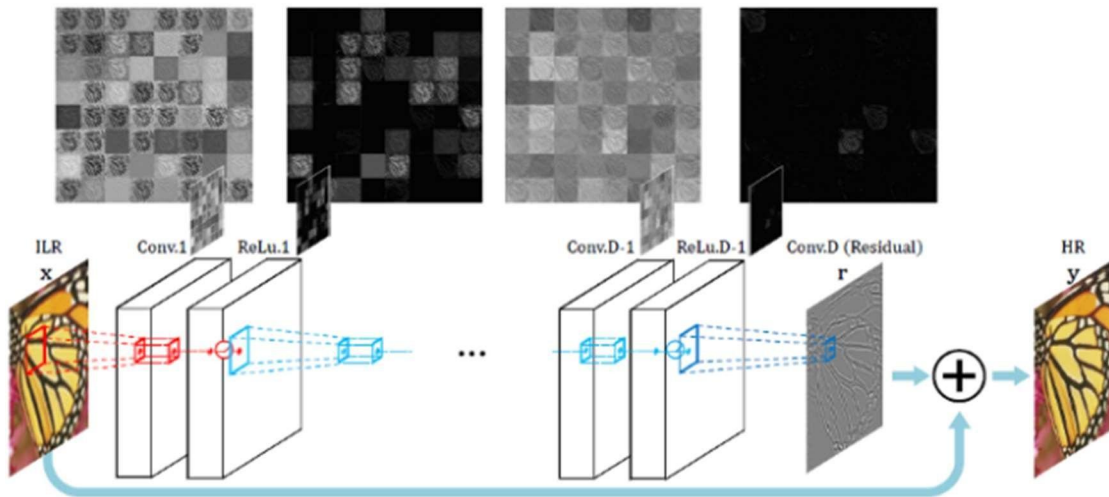


Fig.3.2.1 VDSR Network Architecture

VDSR architecture follows the following steps:

- The LR image is interpolated as ILR image and input to the network.
- The ILR image goes through (D-1) times of Conv and ReLU layers.
- And then followed by a D-th Conv (Conv.D (Residual) in the figure).

Finally, the output is added with the ILR image and obtain the HR image.

These are 64 filters with the size of 3×3 for each conv layer. (VGGNet has addressed the issue of consecutive 3×3 filters helps to obtain larger receptive fields so that we do not require any large filters such as 5×5 and 7×7).

With AI chipsets become popular in the future, VDSR or other state-of-the-art approaches can be applied in real-time for image enlargement, and even applied in video.

3.2.2 FRCNN (FASTER R-CNN)

R-CNN is the first step for Faster R-CNN. It uses search selective to find out the

regions of interests and passes them to a ConvNet. It tries to find out the areas that might be an object by combining similar pixels and textures into several rectangular boxes. The

R-CNN paper uses 2,000 proposed areas (rectangular boxes) from search selective. Then, these 2,000 areas are passed to a pre-trained CNN model. Finally, the outputs (feature maps) are passed to a SVM for classification. The regression between predicted bounding boxes (bboxes) and ground-truth bboxes are computed.

Fast R-CNN moves one step forward. Instead of applying 2,000 times CNN to proposed areas, it only passes the original image to a pre-trained CNN model once. Search selective algorithm is computed base on the output feature map of the previous step. Then, ROI pooling layer is used to ensure the standard and pre-defined output size. These valid outputs are passed to a fully connected layer as inputs. Finally, two output vectors are used to predict the observed object with a softmax classifier and adapt bounding box localisations with a linear regressor.

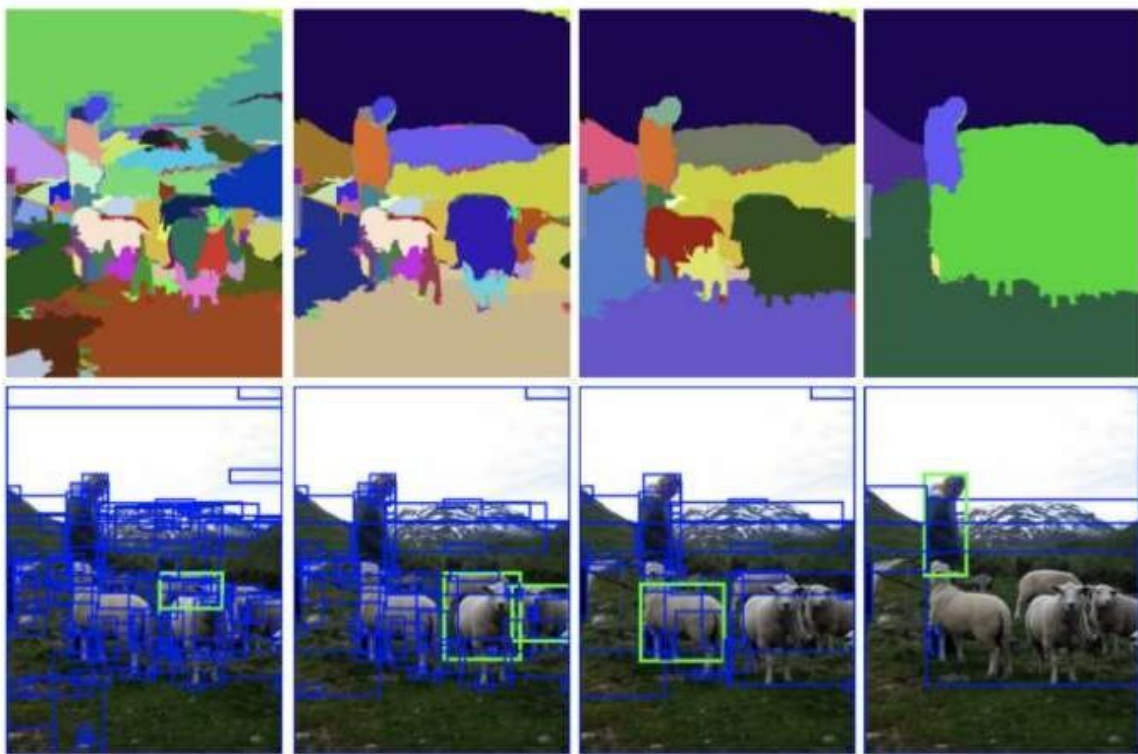


Fig.3.2.2.1. Example of search selective

Faster R-CNN (frcnn for short) makes further progress than Fast R-CNN. Search selective process is replaced by Region Proposal Network (RPN). As the name revealed, RPN is a network to propose regions. For instance, after getting the output feature map from a pre-trained model (VGG-16), if the input image has 600x800x3 dimensions, the output feature map would be 37x50x256 dimensions.

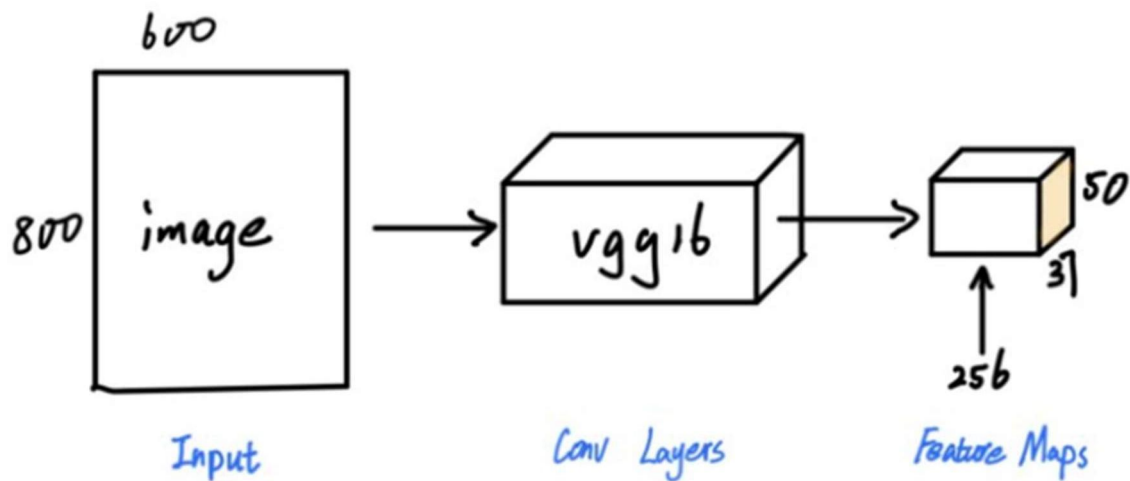


Fig. 3.2.2.2. First step of FRCNN

Each point in 37x50 is considered as an anchor. We need to define specific ratios and sizes for each anchor (1:1, 1:2, 2:1 for three ratios and 128^2 , 256^2 , 512^2 for three sizes in the original image).

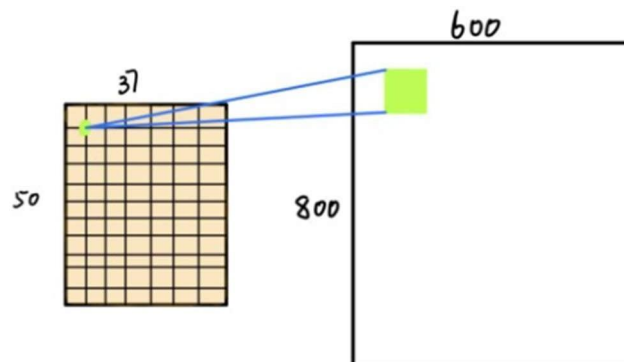


Fig. 3.2.2.3. One anchor projected to the original image.

Next, RPN is connected to a Conv layer with 3x3 filters, 1 padding, 512 output channels. The output is connected to two 1x1 convolutional layer for classification and box- regression (Note that the classification here is to determine if the box is an object or not). Then we go to the second stage of frCNN. Similar to Fast R-CNN, ROI pooling is used for these proposed regions (ROIs). The output is 7x7x512. Then, we flatten this layer with some fully connected layers. The final step is a softmax function for classification and linear regression to fix the boxes' location.

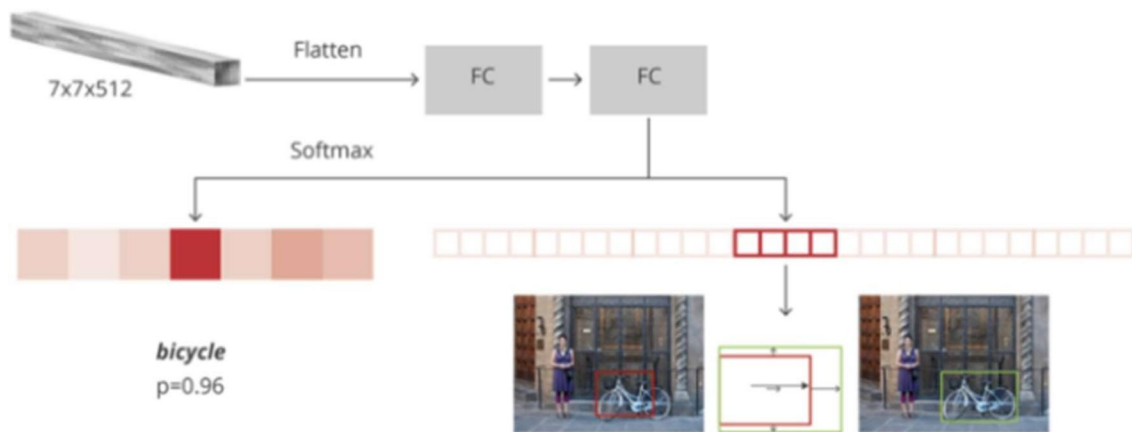


Fig.3.1.6.4. R-CNN Architecture

3.2.3 PSNR (Peak Signal-to-Noise-Ratio)

Peak signal-to-noise ratio (PSNR) is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. Because many signals have a very wide dynamic range, PSNR is usually expressed as a logarithmic quantity using the decibel scale. For color images with three RGB values per pixel, the definition of PSNR is the same except the MSE is the sum over all squared value differences (now for each color, i.e. three times as many differences as in a monochrome image) divided by image size

and by three.

Alternately, for color images the image is converted to a different color space and PSNR is reported against each channel of that color space, e.g., YCbCr or HSL.

PSNR is most commonly used to measure the quality of reconstruction of lossy compression codecs (e.g., for image compression). The signal in this case is the original data, and the noise is the error introduced by compression. When comparing compression codecs, PSNR is an approximation to human perception of reconstruction quality.

Typical values for the PSNR in lossy image and video compression are between 30 and 50 dB, provided the bit depth is 8 bits, where higher is better. For 16-bit data typical values for the PSNR are between 60 and 80 dB. Acceptable values for wireless transmission quality loss are considered to be about 20 dB to 25 dB.

In the absence of noise, the two images I and K are identical, and thus the MSE is zero. In this case the PSNR is infinite (or undefined, see Division by zero).

Although a higher PSNR generally indicates that the reconstruction is of higher quality, in some cases it may not. One has to be extremely careful with the range of validity of this metric; it is only conclusively valid when it is used to compare results from the same codec (or codec type) and same content. Generally, PSNR has been shown to perform poorly compared to other quality metrics when it comes to estimating the quality of images and particularly videos as perceived by humans.

PSNR-HVS is an extension of PSNR that incorporates properties of the human visual system such as contrast perception. PSNR-HVS-M improves on PSNR-HVS by additionally taking into account visual masking. In a 2007 study, it delivered better approximations of human visual quality judgements than PSNR and SSIM by large margin. It was also shown to have a distinct advantage over DCTune and PSNR-HVS. The mean squared error (MSE) for our practical purposes allows us to compare the “true” pixel values of our original image to our degraded image. The

MSE represents the average of the squares of the "errors" between our actual image and our noisy

image. The error is the amount by which the values of the original image differ from the degraded image.

The proposal is that the higher the PSNR, the better degraded image has been reconstructed to match the original image and the better the reconstructive algorithm. This would occur because we wish to minimize the MSE between images with respect the maximum signal value of the image.

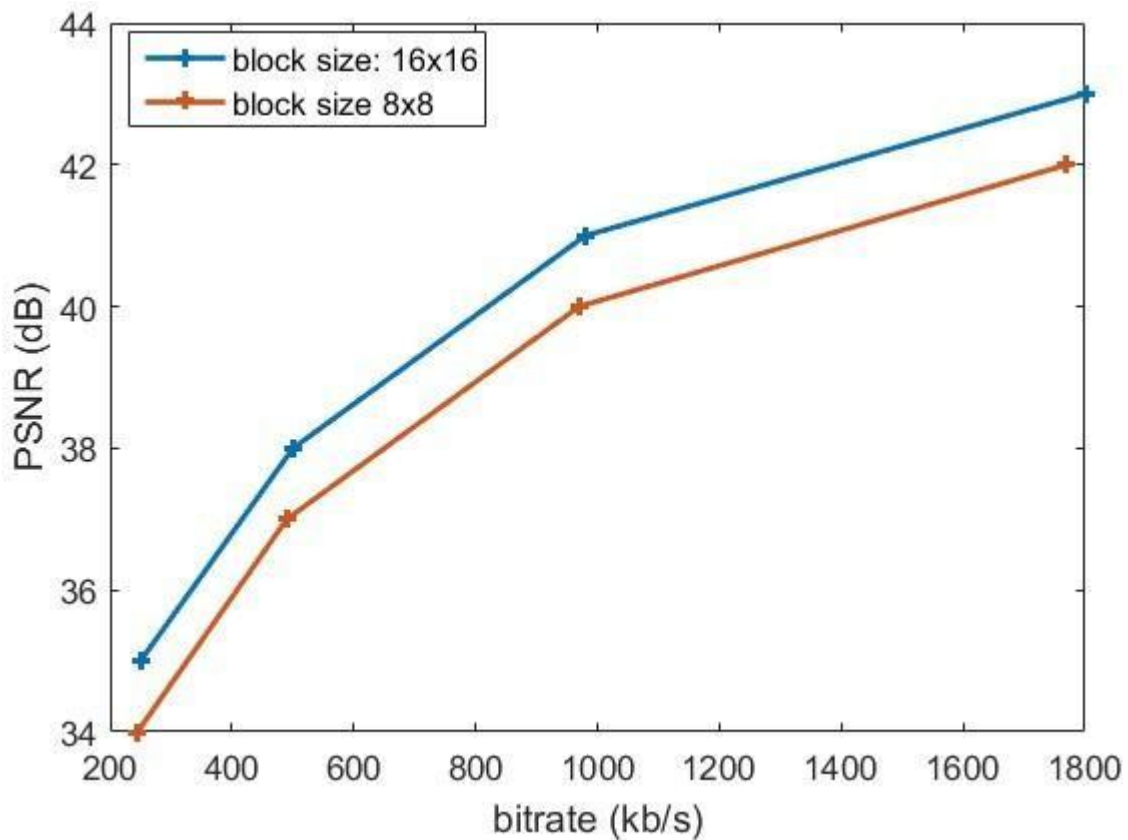


Fig.3.2.3 PSNR Behavior with respect to bitrate

3.2.4 SRCNN (Super Resolution Convolutional Neural Network)

Image super-resolution (SR) is to convert a given low-resolution (LR) image into a corresponding high-resolution (HR) image by an algorithm [1, 2, 3]. In practice, SR

has a wide range of application scenarios and application prospects, such as video, security

and medical imaging where demand more image details. In early, traditional methods of SR, including interpolation, sparse coding and other methods, require rigorous algorithm to extract engineered features. With the development of CNN, Dong et al. present a method in 2014, named SRCNN, using convolutional neural network. SRCNN is the first method to apply CNN to image super-resolution (SR) and achieves the state-of-the-art performance. It fully demonstrates the effectiveness of CNN for SR and is an end-to-end method. End-to-end means inputting a LR image and then outputting a HR image, unnecessary to care about the middle of the conversion process [4,5]. The methods for SR based on CNN, is only needing to give images and the architecture of network for training, which makes these methods have a wider range of applicability.

So, more and more researchers have begun to use deep learning to do the image studies, not only for SR, and have made a lot of progress. In the field of image super-resolution (SR), the depth of convolutional networks have increased from SRCNN's 3 layers to currently 20-30 layers [7, 8, and 9].

Recently many methods using CNN for SR have been proposed. Among them, VDSR shows the best performance. VDSR is proposed by Kim et al. in 2016 and the depth of its network is 20 layers. VDSR uses cascading the same convolutional layers and learning residuals to reduce the super-parameters of networks and speed-up the training. Although the promotion of VDSR for SR is noticeable, but the amount of the network parameters is still huge. If continuing to increase the depth of the network, we will face the serious growth of computing complexity, which makes the training of the network difficult to converge and even effects the performance bad.

As known, huge growth of the amounts of parameters is accompanied by over-fitting. We would like to proposed a strategy to reduce the amount of the

network parameters, while the performance is as good as before. In the field of classification and recognition, deep learning methods have also take place of the traditional methods. And some methods based on CNN have using topology strategy to improve the architecture of

networks, such as GoogLeNet, Res NeXt. Google Net achieved the 1st place in the ILSVRC (ImageNet Large Scale Visual Recognition Competition) classification task and Res Ne Xt secured 2nd place in ILSVRC 2016 classification.

GoogLeNet apply carefully designed topologies that allows for increasing the depth and width of the network while keeping the computational budget constant . Res Ne Xt use simply homogeneous topologies to improve Res Net to get higher classification accuracy. SR and classification are different tasks. And, as far as we know, in the field of image super resolution (SR), so far no article is enhancing the performance of SR by utilizing topology aggregation strategy on the architecture of CNN. But the convolutional layers are all use to extract features and the topology aggregation strategy is just to adjust the CNN internal structure, so topology structure should also work well on SR task. Our experiments shows that suitable topology aggregation structure is also applicable to SR task and get better results.

Contribution In summary, in this paper, we propose a new architecture of the deep convolutional network for single-image super-resolution (SR). Our network is constructed by repeating the same blocks and each block has the same uniform topology aggregation structure. Experiments show that if the depth of our network is 20 layers, as same as VDSR, our model size is smaller than VDSR 1/3 and the performance is as good as VDSR. Moreover, if we set our model size is as same as VDSR's model size, the depth of our network can be increased to 32 layers, and the performance is better than VDSR.

SRCNN is the first method using CNN for SR. SRCNN is related to traditional sparse-coding-based SR methods, which provides a guidance for the design of the network structure. Its network is composed of three convolution layers. And the size of the convolution kernel is $9 * 9$, $1 * 1$, and $5 * 5$ respectively. The speed of SRCNN is faster than traditional methods, because it is an end-to-end method and is

completely running forward. It has shown convolutional networks can get better results than traditional SR method [3, 14, 16]. VDSR, utilizing cascading small filters many times in a deep network

structure and learning residuals, gets the state-of-the-art best performance of SR. Its network is simple and efficient and the depth of the network is 20 layers. Each convolution layer is the same and has the same 3×3 filter size.

The most straightforward way of improving the performance of deep neural networks is by increasing their size. This includes both increasing the depth – the number of levels

– of the network and its width: the number of units at each level. This is as an easy and safe way of training higher quality models, especially given the availability of a large amount of labeled training data [14,17]. Bigger size typically means a larger number of parameters, which makes the training of the network more difficult to convergence and the enlarged network more prone to overfitting, especially if the number of labeled examples in the training set is limited. Our proposed deep convolutional networks method for SR applies topology aggregation strategy to adjust the inner structure of the architecture of VDSR. Our design makes the number of network parameters less and has few super parameters to set, while the performance of SR can be maintained. Moreover, our design makes the depth of the network can be increased. Our network is constructed by stacking several same blocks. Each block has the same topology aggregation structure. In VDSR, the architecture of the network is consisting of 20 same convolutional layers.

The structure is consisting of cascading 3 same convolutional layers. The architecture of VDSR is the same as the structure. And just the network of VDSR has 20 layers. We use a uniform topology aggregation strategy to divide inputs and outputs of the three convolutional layers into multiple branches. The structure has 32 branches and each branch has the same number of inputs and outputs. In VDSR, the number of outputs is 64, kernel size is 3×3 , stride is 1, and pad is 1 and learning residuals between LR image and HR image. These strategies in VDSR are inherited by our network. Converting the network of VDSR to our network, needs two steps.

First step, we keep the first and the last layers and then divide the other 18 layers into 6 same blocks. Second step, we need to use uniform topology aggregation to transfer each block to the structure.

CHAPTER 4

SOFTWARE DEVELOPMENT METHODOLOGY

4.1 DESCRIPTION OF DIAGRAM

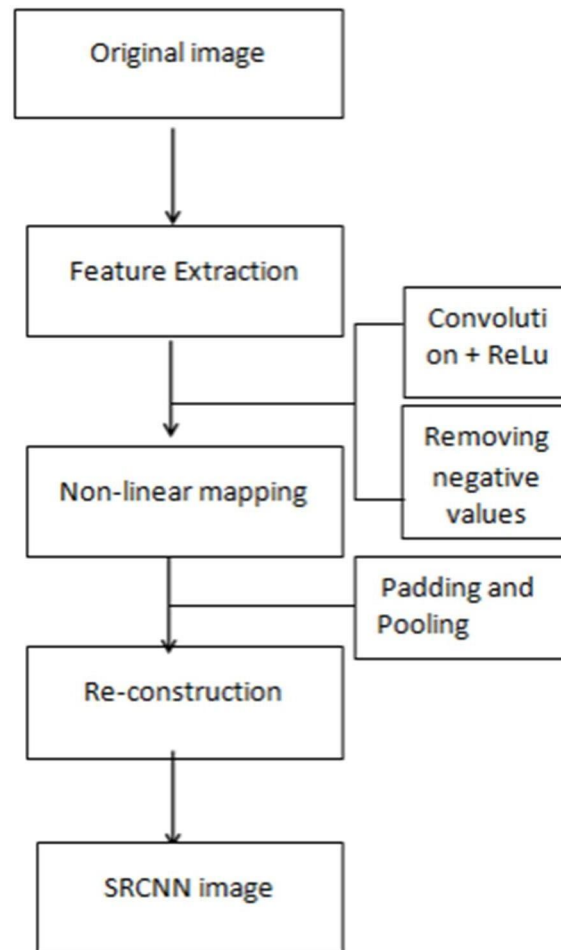


Fig.4.1 Flow Diagram

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, re-cent state-of-the-art methods mostly adopt the example-based strategy. These methods either exploit internal similarities of the same image 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 paper, we show that the afore mentioned 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 modeling 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/post-processing. We name

the proposed model Super-Resolution Convolutional Neural Network (SRCNN)¹. The proposed SRCNN has several appealing properties. First, its structure is intentionally designed with simplicity in mind, and yet provides superior accuracy² compared with state-of-the-

art example-based methods. Figure 1 shows a comparison on an example. Second, with moderate numbers of filters and layers, our method achieves fast speed for practical on-line usage even on a CPU. Our method is faster than a number of example-based methods, because it is fully feed-forward and does not need to solve any optimization problem on usage. Third, experiments show that the restoration quality of the network can be further improved when (i) larger and more diverse datasets are available, and/or (ii) a larger and deeper model is used. On the contrary, larger datasets/models can present challenges for existing example-based methods. Furthermore, while most existing methods are not readily extendable for handling multiple channels in color images, the proposed network can cope with three channels of color images simultaneously to achieve improved super-resolution performance. Overall, the contributions of this study are mainly in three aspects:

- 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/post-processing beyond the optimization.
- We establish a relationship between our deep-learning-based SR method and the traditional sparse-coding-based SR methods. This relationship provides a guidance for the design of the network structure.
- We demonstrate that deep learning is useful in the classical computer vision problem of super-resolution, and can achieve good quality and speed. A preliminary version of this work was presented earlier .

The present work adds to the initial version in significant ways. Firstly, we improve the SRCNN by introducing larger filter size in the non-linear mapping layer, and explore deeper structures by adding non-linear mapping layers. Secondly, we extend the SRCNN to process three color channels (either in YCbCr or RGBcolor 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. We also extend the original experiments from Set5 and Set14 test images to BSD200 [32](200 test images). In

addition, we compare with a number of recently published methods and confirm that our model still outperforms existing approaches using different evaluation metrics.

4.2 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 perform. 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.

- 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.
- Non-linear mapping: this operation non linearly maps each high-dimensional vector onto another high-dimensional vector. Each mapped vector is conceptually the representation of a high-resolution patch. These vectors comprise another set of feature map.
- 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 .

We will show that all these operations form a convolutional neural network. An overview of the network is depicted. Next we detail our definition of each operation.

4.2.1 PATCH EXTRACTION AND REPRESENTATION

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 three. Bicubic interpolation is also a convolutional operation, so it can be

formulated as a convolutional layer. However, the output size of this layer is larger than the input size, so there is a fractional stride. To take advantage of the popular well-optimized implementations such as cuda-convnet, we exclude this “layer” from learning.

the optimization of these bases into the optimization of the network. Formally, our first layer is expressed as an operation $F1$:

$$F1(Y) = \max(0, W1 * Y + B1)$$

where $W1$ and $B1$ represent the filters and biases respectively. Here $W1$ is of a size $c \times f1 \times f1 \times n1$, where c is the number of channels in the input image, $f1$ is the spatial size of a filter, and $n1$ is the number of filters. Intuitively, $W1$ applies $n1$ convolutions on the image, and each convolution has a kernel size $c \times f1 \times f1$. 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 response.

4.2.2 NON-LINEAR MAPPING

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 $n2$ -dimension alone. 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 is:

$$F2(Y) = \max(0, W2 * F1(Y) + B2).$$

Here $W2$ is of a size $n1 \times f2 \times f2 \times n2$, 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 (

$n^2 \times f^2 \times f^2 \times n^2$ parameters for one layer), and thus demands more training time. We will explore deeper structures by introducing additional non-linear mapping layers. The ReLU can be equivalently considered as a part of the second operation (Non-linear mapping), and

the first operation (Patch extraction and representation) becomes purely linear convolution.

4.2.3 RECONSTRUCTION

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:

$$F(Y) = W_3 * F_2(Y) + B_3.$$

Here W_3 is of a size $n_2 \times f_3 \times f_3 \times c$, and B_3 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 W_3 behaves like first projecting the coefficients onto the image domain and then averaging. In either way, W_3 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 weight 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. We detail the relationship in the next section.

4.2.4 RELATIONSHIP TO SPARSE-CODING-BASED METHODS

We show that the sparse-coding-based SR methods can be viewed as a convolutional neural network. In the sparse-coding-based methods, let us consider that an $f_1 \times f_1$ low-resolution patch is extracted from the input image. This patch is

subtracted by its mean, and then is projected onto a (low-resolution) dictionary.

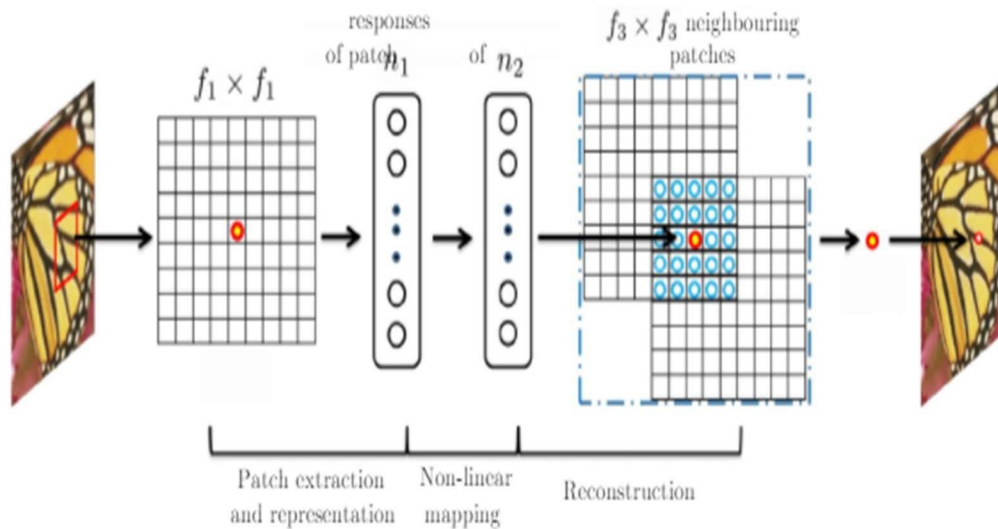


Fig.4.2.4. SRCNN Extraction

If the dictionary size is n_1 , this is equivalent to applying n_1 linear filters ($f_1 \times f_1$) on the input image (the mean subtraction is also a linear operation so can be absorbed). This is illustrated as the left part of .A sparse coding solver will then be applied on the projected n_1 coefficients (e.g., see the Feature-Sign solver). The outputs of this solver are n_2 coefficients, and usually $n_2 = n_1$ in the case of sparse coding.

These n_2 coefficients are the representation of the high-resolution patch. In this sense, the sparse coding solver can be seen as a special case of a non-linear mapping operator, whose spatial support is 1×1 . See the middle part of Figure 4.2.4. 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 $f_2 = 1$, then our non-linear operator can be considered as a pixel-wise fully-connected layer. The above n_2 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 2 feature maps. If the

high-resolution patches used for reconstruction are of size $f_3 \times f_3$, then the linear filters have an equivalent spatial support

of size $f_3 \times f_3$. See the right part of Figure 4.2.4. 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 hyper-parameters. 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 $f_3 = 1$, we are using the center pixel with no averaging). We can also set $n_2 < n_1$ because it is expected to be sparser. Atypical and basic setting is $f_1 = 9, f_2 = 1, f_3 = 5, n_1 = 64$, and $n_2 = 32$ (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 $5 \times 5 = 25$ pixels. This is one of the reasons why the SRCNN gives superior performance.

4.2.5 TRAINING

Learning the end-to-end mapping function F requires the estimation of network parameters $\Theta = \{W_1, W_2, W_3, B_1, B_2, B_3\}$. This is achieved through minimizing the loss between the reconstructed images $F(Y; \Theta)$ and the corresponding ground truth high-resolution images X . Given a set of high-resolution images $\{X_i\}$ and their corresponding low-resolution images $\{Y_i\}$, we use Mean Squared Error (MSE) as the loss function: $L(\Theta) = \frac{1}{n} \sum_{i=1}^n \|F(Y_i; \Theta) - X_i\|_2^2$,

where n is the number of training samples. Using MSE as the loss function favors 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 metrics 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 “hand-crafted” methods. Despite that the proposed model is trained favoring 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

$$\Delta_{i+1} = 0.9 \cdot \Delta_i + \eta \cdot \partial L / \partial W^i, W^{i+1} = W^i + \Delta_{i+1},$$

where $i \in \{1, 2, 3\}$ and i are the indices of layers and iterations, η is the learning rate, and $\partial L / \partial W^i$ 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 (similar to the denoising case). In the training phase, the ground truth images $\{X_i\}$ are prepared as $f_{\text{sub}} \times f_{\text{sub}} \times c$ -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 $\{Y_i\}$, we blur a sub-image by a Gaussian kernel, 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 $((f_{\text{sub}} - f_1 - f_2 - f_3 + 3) \times c)$. The MSE loss function is evaluated only by the difference between the central pixels of X_i 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.

We first investigate the impact of using different data sets on the model performance. Next, we examine the filters learned by our approach. We then explore different

architecture designs of the network, and study the relations between super-resolution performance and factors like depth, number of filters, and filter sizes. Subsequently, we compare our method with recent state-of-the-arts both quantitatively and qualitatively. At last, we extend the network to cope with color images and evaluate the performance on different channels. 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 $f_{sub} = 33$. 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., $f_1 = 9, f_2 = 1, f_3 = 5, n_1 = 64$, and $n_2 = 32$. We use the Set5 as the validation set. We observe a similar trend even if we use the larger Set14 set. The upscaling factor is 3. We use the bicubic interpolation and sparse-coding-based method as our baselines, which achieve an average PSNR value of 30.39 dB and 31.42 dB, respectively. The test convergence curves of using different training sets are shown in Figure 4.2.1. The training time on Image Net 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 back propagations (i.e., 8×10^8), the SRCNN + ImageNet achieves 32.52 dB, higher than 32.39 dB yielded by the original SRCNN trained on 91 images. The results positively indicate that SRCNN performance may be further boosted using a larger and more diverse image training set. Thus, in the following experiments, we adopt the Image Net as the default training set.

4.2.6. LEARNED FILTERS FOR SUPER-RESOLUTION

Interestingly, each learned filter has its specific functionality. For instance, the filters g and h are like Laplacian/Gaussian filters, the filters a-e are like edge detectors at different directions, and the filter f is like a texture extractor.

4.2.7. MODEL AND PERFORMANCE TRADE-OFFS

Based on the basic network settings (i.e., $f_1=9, f_2=1, f_3=5, n_1=64$, and $n_2=32$), we will progressively modify some of these parameters to investigate the best trade-off between performance and speed, and study the relations between performance and parameters.

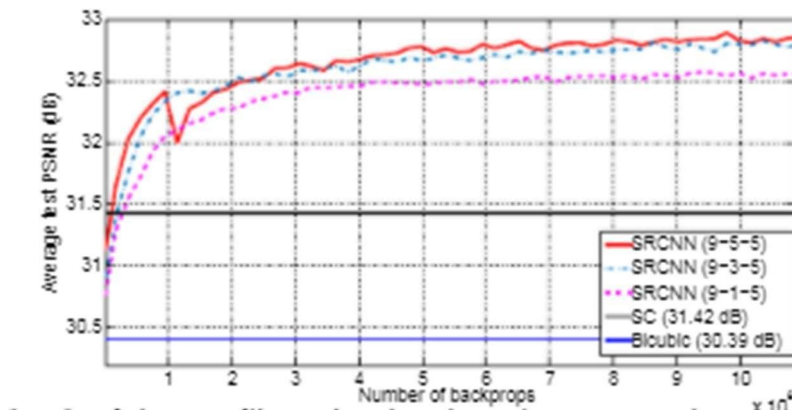


Fig 4.2.7.1 A larger filter leads to better results

In general, the performance would improve if we increase the network width, i.e., adding more filters, at the same depth. We use 'width' to term the number of filters in a layer, following. The term 'width' may have other meanings in the literature cost of running time. Specifically, based on our network default settings of $n_1=64$ and $n_2=32$, we conduct two experiments: (i) one is with a larger network with $n_1=128$ and $n_2=64$, and (ii) the other is with a smaller network with $n_1=32$ and $n_2=16$.

We also train the two models on ImageNet and test on Set5 with an upscaling factor 3. The results observed at 8×10^8 backpropagations are shown in Table 1. It is clear

that superior performance could be achieved by increasing the width. However, if a fast

restoration speed is desired, a small network width is preferred, which could still achieve better performance than the sparse-coding-based method (31.42 dB).

The results of using different filter numbers in SRCNN. Training is performed on ImageNet whilst the evaluation is conducted on the Set5 dataset.

$n_1 = 128$ $n_2 = 64$		$n_1 = 64$ $n_2 = 32$		$n_1 = 32$ $n_2 = 16$	
PSNR	Time (sec)	PSNR	Time (sec)	PSNR	Time (sec)
32.60	0.60	32.52	0.18	32.26	0.05

we examine the network sensitivity to different filter sizes. In previous experiments, we set filter size $f_1 = 9, f_2 = 1$ and $f_3 = 5$, and the network could be denoted as 9-1-5. First, to be consistent with sparse-coding-based methods, we fix the filter size of the second layer to be $f_2 = 1$, and enlarge the filter size of other layers to $f_1 = 11$ and $f_3 = 7$ (11-1-9). All the other settings remain the same with Section 4.1. The results with an upscaling factor 3 on Set5 are 32.57 dB, which is slightly higher than the 32.52 dB reported in Section 4.1. This indicates that a reasonably larger filter size could grasp richer structural information, which in turn lead to better results.

Then we further examine networks with a larger filter size of the second layer. Specifically, we fix the filter size $f_1 = 9, f_3 = 5$, and enlarge the filter size of the second layer to be (i) $f_2 = 3$ (9-3-5) and (ii) $f_2 = 5$ (9-5-5). Convergence curves in Figure 6 show that using a larger filter size could significantly improve the performance. Specifically, the average PSNR values achieved by 9-3-5 and 9-5-5 on Set5 with 8×10^8 backpropagations are 32.66 dB and 32.75 dB, respectively. The results suggest that utilizing neighborhood information in the mapping stage is beneficial. However, the deployment speed will also decrease with a larger filter size.

For example, the number of parameters of 9-1-5, 9-3-5, and 9-5-5 is 8,032, 24,416,

and 57,184 respectively. The complexity of 9-5-5 is almost twice of 9-3-5, but the

performance improvement is marginal. Therefore, the choice of the network scale should always be a trade-off between performance and speed.

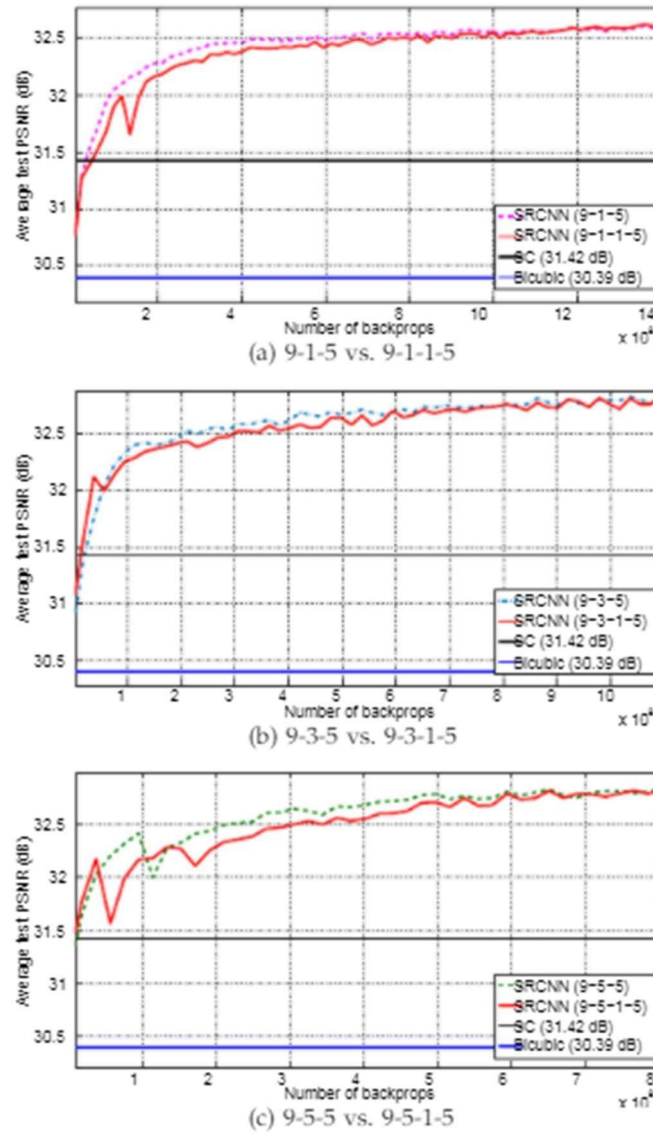


Fig. 4.2.7.2 Comparisons between three-layer and four-layer networks.

Recent study by He and Sun suggests that CNN could benefit from increasing the depth of network 8 moderately. Here, we try deeper structures by adding another non-linear mapping layer, which has $n_{22} = 16$ filters with size $f_{22} = 1$. 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. 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 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 n_{22}

= 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 n_{22}

= 32 and $n_{23} = 16$ 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

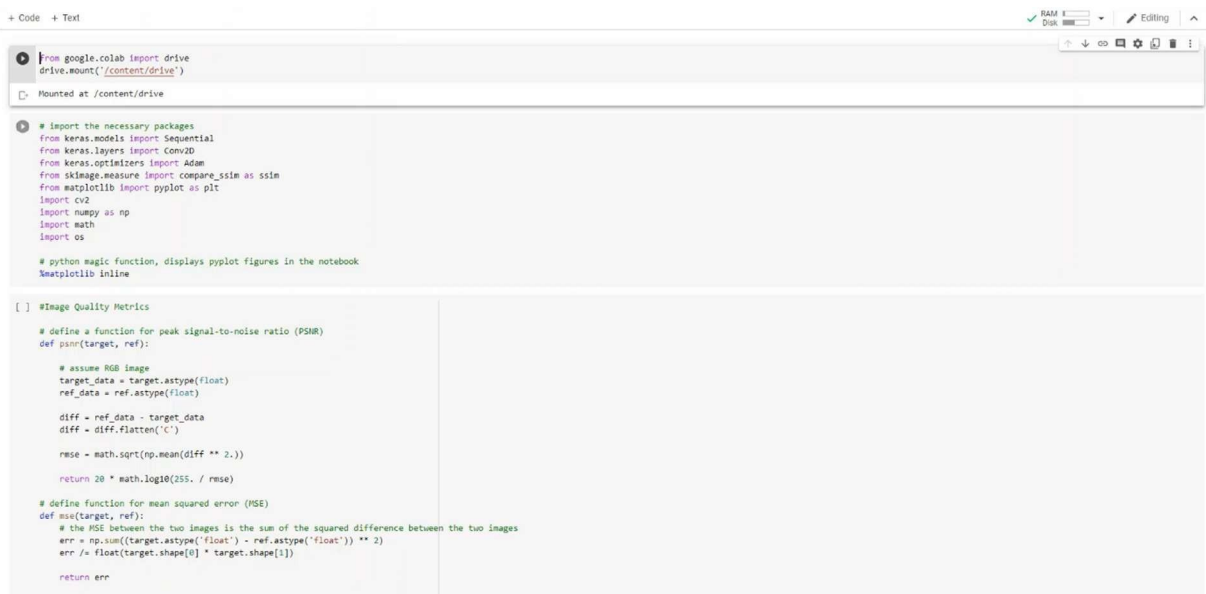
.We also tried to enlarge the filter size of the additional layer to $f_{22} = 3$, and explore two deep structures – 9-3-3-5 and 9-3-3-3. However, from the convergence curves shown, these two networks do not show better results than the 9-3-1-5 network. All these experiments indicate that it is not “the deeper the better” in this deep model for super- resolution. This phenomenon is also observed, where improper increase of depth leads to accuracy saturation or degradation for image classification. Therefore, we still adopt three-layer networks in the system. Running time comparisons of several state-of-the- art methods, along with their restoration performance on Set14.. Note that the processing time of our approach is highly linear to the test image resolution, since all images go through the same number of convolutions. Our method is always a trade-off between performance and speed. To show this, we train three networks for comparison, which are 9-1-5, 9-3-5, and 9-5-5. It is clear that the 9-1-5 network is the fastest, while it still achieves better performance than the next state-of-the-art A+. Other methods are several times or even orders of magnitude slower in comparison to 9-1-5 network. Note the speed gap is not mainly caused by the different MATLAB/C++ implementations; rather, the other methods need to solve

complex optimization problems on usage (e.g., sparse coding or embedding), whereas our method is completely feed-forward. The 9-5-5 network achieves the best performance but at the cost of the running time. The test-time speed of our CNN can be further accelerated in many ways, e.g., approximating or simplifying the trained networks with possible slight degradation in performance.

CHAPTER 5

RESULTS AND DISCUSSION

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.



```
+ Code + Text
RAM 100%
Disk 100%
Editing

1 From google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

2 # Import the necessary packages
from keras.models import Sequential
from keras.layers import Conv2D
from keras.optimizers import Adam
from skimage.measure import compare_ssim as ssim
from matplotlib import pyplot as plt
import cv2
import numpy as np
import math
import os

# python magic function, displays pyplot figures in the notebook
%matplotlib inline

3 [ ] #Image Quality Metrics

# define a function for peak signal-to-noise ratio (PSNR)
def psnr(target, ref):
    # assume RGB image
    target_data = target.astype(float)
    ref_data = ref.astype(float)

    diff = ref_data - target_data
    diff = diff.flatten('C')

    rmse = math.sqrt(np.mean(diff ** 2.))

    return 20 * math.log10(255. / rmse)

# define function for mean squared error (MSE)
def mse(target, ref):
    # the MSE between the two images is the sum of the squared difference between the two images
    err = np.sum((target.astype('float') - ref.astype('float')) ** 2)
    err /= float(target.shape[0] * target.shape[1])

    return err
```

Fig.5.1 Initializing the quality metrics

For each model implemented, we will compute a metric commonly used to measure the quality of reconstruction of lossy compression codecs, called Peak Signal to Noise Ratio (PSNR). This metric is a de-facto standard used in Super Resolution research. It measures how much the distorted image (possibly of lower quality)

deviates from the original high-quality image. PSNR is the ratio of maximum possible pixel value of the image (signal strength) to maximum mean squared error (MSE) between the original image and its estimated version (noise strength), expressed in logarithmic scale.

$$PSNR = 10 \cdot \log_{10} \frac{MAX_I^2}{MSE}$$

The larger the PSNR values, the better the reconstruction, and therefore maximization of PSNR naturally leads to minimizing MSE as the objective function.

The Structural Similarity Index (SSIM) metric extracts 3 key features from an image:

- Luminance
- Contrast
- Structure

The comparison between the two images is performed on the basis of these 3 features. Luminance is measured by averaging over all the pixel values. Its denoted by μ (Mu) and the formula is

$$\mu_x = \frac{1}{N} \sum_{i=1}^N x_i.$$

Contrast is measured by taking the standard deviation (square root of variance) of all the pixel values. It is denoted by σ (sigma) and represented by the formula

$$\sigma_x = \left(\frac{1}{N-1} \sum_{i=1}^N (x_i - \mu_x)^2 \right)^{\frac{1}{2}}.$$

The SSIM score is given by,

$$\text{SSIM}(\mathbf{x}, \mathbf{y}) = [l(\mathbf{x}, \mathbf{y})]^\alpha \cdot [c(\mathbf{x}, \mathbf{y})]^\beta \cdot [s(\mathbf{x}, \mathbf{y})]^\gamma$$

$\alpha > 0, \beta > 0, \gamma > 0$ denote the relative importance of each of the metrics.

$$\text{SSIM}(\mathbf{x}, \mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}.$$

Y only: this is our baseline method, which is a single-channel ($c=1$) network trained only on the luminance channel.

- YCbCr: training is performed on the three channels of the YCbCr space.
- Y pre-train: first, to guarantee the performance on the luminance channel, we only use the MSE of the luminance channel as the loss to pre-train the network. Then we employ the MSE of all channels to fine tune the parameters.
- CbCr pre-train: we use the MSE of the chrominance channels as the loss to pre-train the network, then fine tune the parameters on all channels.
- RGB: training is performed on the three channels of the RGB space.

For each model implemented in SRCNN, we will evaluate the quality of an image which is commonly used to measure loss of feature of an image .which gives more information about reconstruction loss, and quality of the model by commonly used image quality metrics, called as Peak Signal to Noise Ratio (PSNR). These are one of

the most common metrics we will be using in different image restoration techniques. It metrics show the clear idea about the image textures where the distorted image (possibly of lower quality) has the poorer rate compared with the original high-quality image which

we are used for reference. Basically, Peak signal to noise ratio provides an insight on the obtained image which is measured as the ratio of maximum pixel value that are accompanied in the particular region of an image (signal strength). which has relation to maximum mean squared error (MSE) and its estimated version of an image (noise strength) that is going to obtain, which is normally expressed in logarithmic scale. The higher the PSNR values, the better the image reconstruction is done, and therefore with increasing the peak signal to noise ratio we can reduce the mean square value which is the objective function. The Comparison between images is done by using Luminance, Contrast and Structure. Whereas Luminance can be

butterfly_GT.bmp PSNR: 24.782076560337416 MSE: 648.6254119873047 SSIM: 0.8791344763843051	woman_GT.bmp PSNR: 29.326236280817465 MSE: 227.812729498164 SSIM: 0.9335397280466592
face.bmp PSNR: 30.99220650287191 MSE: 155.23189718546524 SSIM: 0.8008439492289884	flowers.bmp PSNR: 27.454504805386144 MSE: 350.55093922651935 SSIM: 0.8697286286974628
bird_GT.bmp PSNR: 32.896644728720005 MSE: 100.12375819830247 SSIM: 0.9533644866026473	baboon.bmp PSNR: 22.157084083442548 MSE: 1187.1161333333334 SSIM: 0.629277587900277
head_GT.bmp PSNR: 31.020502848237534 MSE: 154.2237755102041 SSIM: 0.8011121330733371	zebra.bmp PSNR: 27.909840639329513 MSE: 315.6585459528818 SSIM: 0.8911656209329116
comic.bmp PSNR: 23.799861502225532 MSE: 813.2338836565096 SSIM: 0.8347335416398209	baby_GT.bmp PSNR: 34.371806409661986 MSE: 71.28874588012695 SSIM: 0.9356987872724932
foreman.bmp PSNR: 30.14456532664372 MSE: 188.6883483270202 SSIM: 0.933268417388899	lenna.bmp PSNR: 31.47349297867539 MSE: 138.94800567626953 SSIM: 0.8460989200521499
coastguard.bmp PSNR: 27.161600663887082 MSE: 375.00887784090907 SSIM: 0.756950063354931	pepper.bmp PSNR: 29.88947161686106 MSE: 200.1033935546875 SSIM: 0.8357937568464359

measured by using average of pixel values presents in an image.

Fig.5.2 PSNR Values for Degraded images

Degraded Image:
PSNR: 32.0448352333889
MSE: 121.81985893148101
SSIM: 0.9383957200607694

Reconstructed Image:
PSNR: 36.146986790063245
MSE: 47.369951119115
SSIM: 0.9624923850773585

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:30: UserWarning: DEPRECATED: skimage.measure.compare_ssim has been moved to skimage.metrics



Fig.5.3 Image comparison before and after implementing the model

Degraded Image:
PSNR: 27.248686459559124
MSE: 367.56400047398984
SSIM: 0.8690622024599293

Reconstructed Image:
PSNR: 29.660220614739448
MSE: 210.95001184974524
SSIM: 0.8989099578687939



Fig.5.4 Image comparison before and after implementing the model

We have successfully retracted the High quality images from the degraded images using SRCNN, and we have measured the quality of images by using quality metrics like PSNR,MSE,SSIM.

CHAPTER 6

CONCLUSION AND FUTURE WORK

CONCLUSION

Using SCRNN we have restored the High Quality images from degraded image which has wide range of demand in image processing industry from Satellite mapping during Disaster, Crop Monitoring and Self-driving cars. This would give more advantage for space industry and Governments to act in time on disasters. Thus, the Converting the degraded images to high quality images using convolutional neural network we have achieved our objective of our project.

FUTURE WORK

We have demonstrated the effectiveness of the topology aggregation structure for image super resolution. With this design, the networks for SR can be go deeper and the model size is smaller. We have shown our method outperform the state-of-the-art methods for image super-resolution. In the future we will we will optimize the model by reducing the area of patch extraction by taking the are which is needed to optimize.

REFERENCES

- [1]. Chen, Q., Huang, J., Feris, R., Brown, L.M., Dong, J., Yan, and S.: Deep domain adaptation for describing people based on fine-grained clothing attributes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5315–5324 (2015)
- [2]. Denton, E.L., Chintala, S., Fergus, R., et al.: Deep generative image models using a laplacian pyramid of adversarial networks. In: Advances in Neural Information Processing Systems. pp. 1486–1494 (2015)
- [3]. Dosovitskiy, A., Tobias Spring Enberg, J., Brox, and T.: Learning to generate chairs with convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1538–1546(2015)
- [4]. Eysenck, M.W.: Fundamentals of cognition. Psychology Press East Sussex, UK, USA and Canada (2006)
- [5]. Ganin, Y., Lempitsky, V.: Unsupervised domain adaptation by backpropagation. In: Proceedings of The 32nd International Conference on Machine Learning (2015)
- [6]. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems. pp. 2672–2680 (2014)
- [7]. Gopalan, R., Li, R., Chellappa, R.: Domain adaptation for object recognition: An unsupervised approach. In: Computer Vision (ICCV), 2011 IEEE International Conference on. pp. 999–1006. IEEE (2011)
- [8]. Gregor, K., Danihelka, I., Graves, A., Rezende, D., Wierstra, D.: Draw: A recurrent neural network for image generation. In: Proceedings of the 32nd International Conference on Machine Learning. pp. 1462–1471 (2015)
- [9]. Hinton, G.E., Salakhutdinov, and R.R.: Reducing the dimensionality of data with neural networks. Science 313(5786), 504–507 (2006)
- [10]. Huang, J., Feris, R.S., Chen, Q., Yan, S.: Cross-domain image retrieval with a dual

attribute-aware ranking network. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1062–1070 (2015)

- [11]. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
- [12]. Kulis, B., Saenko, K., Darrell, T.: What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. pp. 1785–1792. IEEE (2011)
- [13]. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., and Hubbard, W., Jackel, L.D.:
Backpropagation applied to handwritten zip code recognition. Neural computation 1(4), 541–551 (1989)
- [14]. Mathieu, M., Couprie, C., LeCun, Y.: Deep multi-scale video prediction beyond mean square error. arXiv preprint arXiv:1511.05440 (2015)
- [15]. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)
- [16]. Oquab, M., Bottou, L., Laptev, I., Sivic, and J.: Learning and transferring mid-level image representations using convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1717–1724 (2014)
- [17]. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
- [18]. Razavian, A., Azizpour, H., Sullivan, J., Carlsson, and S.: Cnn features off-the-shelf: an astounding baseline for recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 806–813 (2014)
- [19]. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision 115(3), 211–252 (2015)
- [20]. Saenko, K., Kulis, B., Fritz, M., Darrell, T.: Adapting visual category models to new domains. In: Computer Vision–ECCV 2010, pp. 213–226. Springer (2010)

APPENDIX

A. Plagiarism Report



Plagiarism Scan Result Report

Scanned Text

Input File	SRCNN_Paper_CSE-455.pdf
Author	n/a
Scan Date	2021-03-05
Project	Sonstige Tests
Remarks	n/a

Result Overview

Copied/Total Words: 131/2131
Suspicious Sources: 8
Accepted Sources: 0
Excluded Sources: 0

Percentage of Copied Words / Total Words



Overview of Matching Phrases



Legend and Explanations

Original Text	Text without any identified relevant online or library sources.
<u>Suspicious source</u>	Text containing identified matches in online or library sources. The text might have been paraphrased.
<u>Accepted source</u>	Text containing identified matches in online or library sources. The text might have been paraphrased.

Searched Sources and Settings

Maximum phrase distance	125	Searched in online sources	Yes
Minimum characters per phrase	125	Searched in library sources	Yes
Minimum characters per phrase	50	Minimal Similarity	40%

B. PAPER

CONVERTING DEGRADED IMAGES TO HIGH RESOLUTION IMAGES USING CONVOLUTIONAL NEURAL NETWORK

¹A.Sivasangari

Dept of IT, Sathyabama Institute
of Science and Technology
Chennai, India
sivasangarikavya@gmail.com

²Harsha vardhan G

Dept of CSE, Sathyabama Institute
of Science and Technology
Chennai India
gaddam.harsha21@gmail.com

³Jujjavarapu Dileep kumar

Dept of CSE, Sathyabama Institute
of Science and Technology
Chennai India
dileepchoudary82@gmail.com

Abstract— In uncertain times like natural disaster satellite imagery plays a key role in navigating the places which affected the most and also to monitor the fore coming situation, to get better insights from the images, it need be convinible that the quality of the image meets the quality that is required. Presently, we have wide range of approaches to restore the good quality image from distorted image using different Deep learning techniques. To solve the problem with degraded images, we have different methods like Full resolution convolutional neural network, Deep resolution convolution neural network and Very deep resolution neural network and PSNR (Peak signal to noise ratio). Among all PSNR provide better colors. But PSNR absences in providing better quality of textures. To overcome this problem, we have proposed a model to restore the quality of an image using SRCNN (Super Resolution Convolution Neural Network). Super resolution is one of the image restoration technique where we will split each layer and then will perform end-to-end mapping in each layer to obtain a high-quality image from a diminished one. With this proposed system we will restore the better texture for an image with input as degraded image and yields the similar scope image as output with High tenacity.

Keywords— Non-linear mapping, Image processing, Quality metrics, model buliding, image analysis.

I. INTRODUCTION

Machine Learning (ML) is one of the prominent fields in technology which is delivering the solutions to most complex problems we face in our day-to-day life. Machine Learning is a branch of Artificial Intelligence (AI) that machine learns by continues learning by data as input and provides the solution to the user for the problem to understand well. In Machine Learning, model train based on huge chunk of data with the help of algorithms without being explicitly programmed. Today, the applications of machine learning are vast and it's been part of our life. The search engines like DuckDuckGo, Bing and Ecosia, which get back to our queries instantly finding the relevant content among billions of web pages are powered by machine learning. Currently, video streaming platforms like Netflix, Hulu, Amazon has huge advantage with machine learning which recommend the user what they like most and how they watch the content on the platforms. Machine learning also increased productivity of every user to make use of the technology with voice assistants like as Google Assistant, Siri, Alexa and Cortana to schedule the meeting, preparing to-do list

and search quick queries. All those applications available today are trained with huge datasets to provide better experiences. Machine Learning algorithms are furtherly divided into three groups Supervised, Semi-Supervised, Unsupervised and Reinforcement learning. Deep learning is one among different types of machine learning (ML) fields that replicates the human's brain that learns from huge amount of data. Deep learning has a significant advantage in data science, which comprises statistics and predictive modelling. It is enormously helpful to data scientists and data analysts who are tasked with aggregating, evaluating and interpreting large amounts of data. Deep learning models are charming in many challenging problems like self-driving vehicles and social media Analytics, deep learning solves the complex problems quicker and easier. Convolutional neural network (CNN), is one among different classes of neural networks that crack out to be foremost innumerable computer vision tasks, and is fascinating interest from wide variety of fields, including Self driving cars and Space industry. CNN is build based on numerous building blocks of network which is basically called as node, in which each node in the network has weights and follows both backward and forward propagation.

II. RELATED WORK

Kingma, Diederik P and Welling, Max. have proposed[4] a model for improving the quality of the image using fidelity measurements, with this model there is lot of improvement in retracting the similar quality as the original image. D. C. Ciresan, U. Meier and J. Schmidhuber have came up with four state evaluation of network[1], when a source image is given as an input the network improves the quality of an degraded in each stage of the process. Instead of working on entire image R. Timothee, R. Rothe, and L. Van Gool [7] proposed to extract the area where the textures of an image is degraded. with this idea the complexity of the network increases but the efficiency of the model seen greater significance.

III. PROPOSED MODEL

Image restoration (IR) is one of the most used in the industry to meet the requirements to get better insights from high quality (HQ) image which is extracted from the distorted observation. Although Deep Learning is disreputably acknowledged for

being able to pull off better insights from the images, we use many IR techniques for different problem to solve the problem and to obtain the more information from the image. Converting High Resolution image from degraded image is always a thought-provoking problem because degraded image input corresponds to a crop of possible high-resolution images and in SRCNN we will map the high resolution with degraded image by reinforcing the feature area. Super resolution is one of the image restoration technique where we will split each layer and then the network will perform end-to-end mapping in each layer to obtain a high-quality image from the diminished one. As a result, we can remove the negative values from the patch, so that we can restore the textures of an image. To evaluate the performance of this model, we use some of the image quality metrics, image quality metrics are significant to understand the changes of an image. Image quality measurement's like, mean squared error (MSE), and structural similarity (SSIM) index which is based on maximum of Mean square error, peak signal to noise ratio (PSNR) which gives a patch signal to noise ratio value. We will also be using an opensource image processing library called OpenCV to modify and process the image. Also, there is recurrent altering of our images.

$$O((f_1^2 n_1 + n_1 f_2^2 n_2 + n_2 f_3^2) S_{HR})$$

Fig.1: Computational Complexity of SRCNN

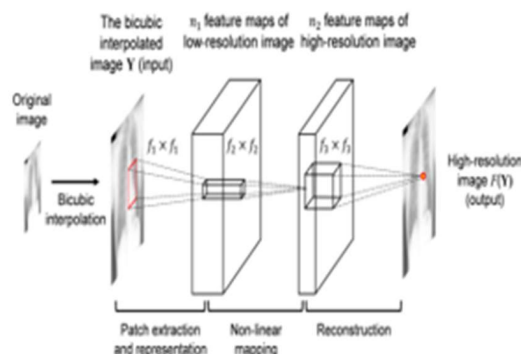


Fig.2. System Architecture

We normally use Convolutional neural network for the image classification problems which has advantage over another neural network. Super Resolution is one among the deep learning techniques which is commonly used for super resolution of images where it will map direct end-to-end mapping between the Low resolution(Degraded) and Super resolution images. Which had wide advantage in super resolution (SR) which is one of the most shared and common one we face in image restoration and image classification techniques. In SRCNN, normally the neural network is not deep as it represents. Super resolution convolutional neural which is

broadly classified into subtypes named as, non-linear mapping, patch extraction, representation, and reconstruction.

A. Feature Extraction

Feature Extraction is the first step we will follow in the super resolution. In the process instead of extracting the entire image which will changes the whole textures of an image, we just extract the area which has distorted and remove the negatives values in the rectified linear unit. Feature extraction which is represented by the formula in the fig2, where X represent the high-resolution image ground truth and Y represent the unsampled version of degraded image. Outliers are represented as $F_1(Y)$

$$F_1(Y) = \max(0, W_1 * Y + B_1)$$

Fig.3: Patch Extraction

Size of W_1 was given by the product of all the quality of channels, filter sizes and dimensional bias vector. where c is represented as quantity of channels of the image, whereas f_1 is represented as the filter size, and number of filters is represented by n_1 . B_1 is used to increase the degree of freedom by 1 which is a n_1 - dimensional bias vector.

B. Non-linear mapping

Non-linear mapping is a second stage of super resolution where we focus on removing negative values

$$F_2(Y) = \max(0, W_2 * F_1(Y) + B_2)$$

The second layer

Fig.4: Non-linear mapping

Size of W_2 was given by the product of all the quality of channels, filter sizes and dimensional bias vector. It is a mapping between n_1 -dimensional vector with the n_2 -dimensional vector. When n_1 -dimensional vector is greater than n_2 -dimensional vector, we can imagine that which is very similar to PCA but in a non-linear way.

In non-linear mapping will map the network will 1×1 where each and every negative value are removed and we will focus on retrieving the texture which is nearby so the image would retain the original image clarity

C. Reconstruction

After mapping, we need to reconstruct the image to derive our target. Hence, we do convolution vector again.

$$F(Y) = W_3 * F_2(Y) + B_3$$

The third layer

Fig.5: Reconstruction

D. Loss Function

Loss function is one of the key metrics in evaluating the quality. Loss function gives the clear view about the quality of the image. It is given by the equation,

$$L(\theta) = \frac{1}{n} \sum_{i=1}^n \|F(Y_i; \theta) - X_i\|^2$$

Loss function

Fig.6: Loss Function

E. Relationship with Sparse Coding

In Sparse Coding (SC), during the process of convolution, the input image which is of poor quality is converted by size of the filter (f1) and projected into a n1-dimensional bias vector. When n1 dimensional bias is equal to the n2 dimensional bias usually is the case of Sparse coding. The mapping is done between n1 and n2 which is a three-dimensional bias vector without reducing the dimensionality. This process is similar to end-to-end mapping where we map the degraded resolution vector with good resolution vector. Later, each patch is restored by filter size (f3). And patches which are overlaid are made an average instead of adding with different convolution weights.

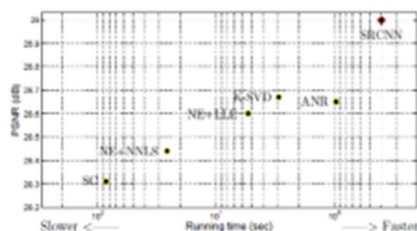


Fig.7: PSNR against time

SRCNN gives the better quality images which has less feature loss and also SRCNN shows the significant performance with faster implementation.

IV. PERFORMANCE ANALYSIS

For each model implemented in SRCNN, we will evaluate the quality of an image which is commonly used to measure loss of feature of an image .which gives more information about reconstruction loss, and quality of the model by commonly used image quality metrics, called as Peak Signal to Noise Ratio

(PSNR). These are one of the most common metrics we will be using in different image restoration techniques. It metrics show the clear idea about the image textures where the distorted image (possibly of lower quality) has the poorer rate compared with the original high-quality image which we are used for reference. Basically, Peak signal to noise ratio provides an insight on the obtained image which is measured as the ratio of maximum pixel value that are accompanied in the particular region of an image (signal strength). which has relation to maximum mean squared error (MSE) and its estimated version of an image (noise strength) that is going to obtain, which is normally expressed in logarithmic scale[4]. The higher the PSNR values, the better the image reconstruction is done, and therefore with increasing the peak signal to noise ratio we can reduce the mean square value which is the objective function. The Comparison between images is done by using Luminance, Contrast and Structure

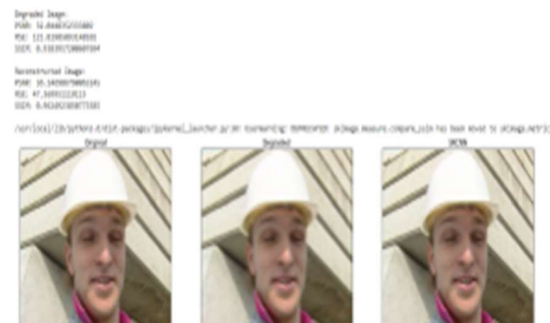


Fig8: Image comparison of model

By using the three metrics for every image we are measuring the quality of the image that have been restored



Fig9: Image comparison of model

The SSIM score is given by,

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

Fig10: SSIM Score

Set14 [26] images	scale	Baseline		RC [26]		K-PSVD [28]		SRCNN [12]		SRCNN [14]		ANF [20]		SRCNN [21]	
		Scale	Precision	Time	Precision	Time	Precision	Time	Precision	Time	Precision	Time	Precision	Time	
baboon	3	23.21	-	23.47	126.3	23.32	3.6	23.49	29.0	23.55	5.6	23.56	1.1	23.60	0.40
baboon	3	26.25	-	26.30	127.9	26.76	3.5	26.67	47.6	26.74	9.8	26.80	1.7	26.88	0.70
bridge	3	24.40	-	24.82	152.7	25.02	3.3	24.86	30.4	24.98	5.9	25.01	1.1	25.07	0.44
camargue	3	26.55	-	27.02	35.6	27.13	1.3	27.00	11.6	27.07	2.6	27.08	0.43	27.20	0.17
comic	3	23.12	-	23.90	34.5	23.96	1.2	23.83	11.0	23.98	2.0	24.01	0.43	24.09	0.15
face	3	22.92	-	23.11	29.4	23.23	1.1	23.45	9.3	23.56	1.7	23.62	0.36	23.74	0.13
flowers	3	27.23	-	28.35	76.4	28.43	2.3	28.21	30.2	28.38	4.0	28.45	0.81	28.97	0.30
foreman	3	31.18	-	32.60	35.9	32.19	1.3	32.87	10.8	33.21	2.2	33.23	0.44	33.35	0.17
horse	3	31.88	-	32.64	48.4	33.00	3.3	32.82	29.3	33.01	4.0	33.08	1.1	33.39	0.44
lion	3	27.81	-	27.78	111.2	27.90	3.4	27.72	29.5	27.87	6.1	27.92	1.1	28.18	0.44
monkey	3	29.43	-	30.71	112.1	31.10	4.9	30.76	43.3	30.95	8.8	31.00	1.6	32.39	0.60
pepper	3	32.30	-	33.32	66.3	34.07	3.3	33.56	28.9	33.80	4.6	33.82	1.1	34.35	0.44
pp3	3	23.71	-	24.08	96.1	25.23	4.0	24.81	36.0	24.94	7.8	25.03	1.4	26.02	0.58
refers	3	26.63	-	27.05	114.4	28.49	2.9	28.12	26.3	28.31	5.5	28.43	1.0	28.87	0.38
average	3	27.54	-	28.31	84.68	28.07	2.55	28.44	25.87	28.60	5.31	28.63	0.97	29.00	0.30

Fig11: SRCNN Performance Score

Set14 28 image	Scale	Baseline		RC [26]		K-PSVD [28]		SRCNN [12]		SRCNN [14]		SRCNN [20]		SRCNN [21]	
		Scale	Precision	Time	Precision	Time	Precision	Time	Precision	Time	Precision	Time	Precision	Time	
baboon	3	23.07	-	-	38.25	7.8	38.05	48.6	38.31	13.6	38.44	3.1	38.50	0.38	-
baboon	2	26.81	-	-	39.03	2.2	39.41	22.5	40.00	4.2	40.04	0.62	40.66	0.14	-
bridge	2	27.43	-	-	39.65	1.8	39.03	18.4	39.38	3.3	39.48	0.59	39.90	0.22	10.1
camargue	2	24.86	-	-	35.50	2.1	35.48	19.2	35.63	3.8	35.66	0.57	35.64	0.13	-
woman	2	22.14	-	-	34.49	2.1	34.24	19.3	34.52	3.8	34.55	0.57	34.94	0.13	-
average	3	33.68	-	-	35.78	5.03	35.43	20.23	35.77	5.74	35.83	0.97	36.94	0.18	-
baboon	4	33.91	-	34.29	76.0	35.08	3.3	34.77	26.3	35.06	6.0	35.13	1.3	35.01	0.28
baboon	3	32.58	-	34.11	36.4	34.37	1.0	34.38	8.9	34.56	1.9	34.61	0.39	34.88	0.14
bridge	4	24.04	-	25.58	26.8	25.94	0.81	25.61	7.0	25.75	1.4	25.90	0.31	26.05	0.13
camargue	4	32.88	-	33.17	21.3	33.56	1.0	33.45	8.2	33.60	1.7	33.68	0.31	33.55	0.13
woman	4	29.56	-	29.94	25.1	30.37	1.0	29.89	8.7	30.22	1.9	30.33	0.37	30.02	0.13
average	3	30.39	-	31.42	135.92	31.80	1.42	31.60	12.21	31.84	2.58	31.92	0.54	32.00	0.18
baboon	4	31.78	-	33.00	2.4	33.81	18.2	33.99	3.8	34.08	0.81	34.09	0.38	34.08	0.13
baboon	4	30.18	-	31.71	0.68	31.51	4.7	31.72	1.1	31.82	0.27	31.88	0.14	31.88	0.13
bridge	4	22.19	-	23.37	0.50	23.30	3.8	23.38	0.80	23.52	1.4	23.64	0.09	23.70	0.10
camargue	4	31.29	-	32.21	0.88	32.10	4.5	32.24	1.1	32.29	0.27	32.33	0.13	32.33	0.13
woman	4	26.45	-	27.80	0.66	27.61	4.3	27.72	1.1	27.80	0.28	28.21	0.13	28.21	0.13
average	4	28.42	-	29.60	0.88	29.47	6.71	29.61	1.56	29.69	0.38	30.00	0.18	30.00	0.18

Fig12: SRCNN relativity Score

CONCLUSION

In this paper, Using SRCNN we have restored the High Quality images from degraded image which has wide range of demand in image processing industry from Satellite mapping during Disaster, Crop Monitoring and Self-driving cars. This would give more advantage for space industry and Governments to act in time on disasters. Thus, the Converting the degraded images to high quality images using convolutional neural network we have achieved our objective of our project.

REFERENCES

- [1] D. C. Ciresan, U. Meier, J. Masci, L. Maria Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in IJCAI Proceedings-International Joint Conference on Artificial Intelligence, vol. 22, no. 1, Barcelona, Spain, 2011, p. 1237.
- [2] Graves, Alex, Mohamed, Abdel-rahman, and Hinton, Geoffrey. Speech recognition with deep recurrent neural networks. In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, pp. 6645-6649. IEEE, 2013.
- [3] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026-1034.
- [4] Kingma, Diederik P and Welling, Max. Auto-Encoding Variational Bayes. In The 2nd International Conference on Learning Representations (ICLR), 2013.
- [5] Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, 2012, pp. 1097-1105.
- [6] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in Proceedings of the 25th international conference on Machine learning. ACM, 2008, pp. 160-167.
- [7] R. Timofte, R. Rothe, and L. Van Gool, "Seven ways to improve example-based single image super resolution," in Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on, IEEE, 2016, pp. 1865-1873.
- [8] S.C. Park, M. K. Park, and M. G. Kang, "Super resolution image reconstruction: a technical overview," IEEE signal processing magazine, vol.20, no.3, pp. 21-36, 2003.
- [9] W. Liu, D. Lin and X. Tang, "Hallucinating faces: Tensorpatch super-resolution and coupled residue compensation", Proc. IEEE Conf. Comput. Vis. Pattern Recognit., vol. 2, pp. 478-484, 2006.
- [10] V.L. LeCun, Y. Bengio, and G. Hinton, "Deep learning," nature, vol.521, no.7553, p.436, 2015.

D. Source Code

```
# import the necessary packages
from keras.models import Sequential
from keras.layers import Conv2D
from keras.optimizers import Adam
from skimage.measure import compare_ssim as ssim
from matplotlib import pyplot as plt
import cv2
import numpy as np
import math
import os

# python magic function, displays pyplot figures in the notebook
%matplotlib inline

#Image Quality Metrics

# define a function for peak signal-to-noise ratio (PSNR)
def psnr(target, ref):

    # assume RGB image
    target_data = target.astype(float)
    ref_data = ref.astype(float)

    diff = ref_data - target_data
    diff = diff.flatten('C')

    rmse = math.sqrt(np.mean(diff ** 2.))

    return 20 * math.log10(255. / rmse)

# define function for mean squared error
(MSE) def mse(target, ref):
    # the MSE between the two images is the sum of the squared difference between the two images
    err = np.sum((target.astype('float') - ref.astype('float')) ** 2)
    err /= float(target.shape[0] * target.shape[1])
```

```
return err
```

```

# define function that combines all three image quality metrics
def compare_images(target, ref):
    scores = []
    scores.append(psnr(target, ref))
    scores.append(mse(target, ref))
    scores.append(ssim(target, ref, multichannel =True))

    return scores

# prepare degraded images by introducing quality distortions via resizing

def prepare_images(path, factor):

    # loop through the files in the
    directory for file in os.listdir(path):

        # open the file
        img = cv2.imread(path + '/' + file)

        # find old and new image dimensions
        h, w, _ = img.shape
        new_height = int(h // factor)
        new_width = int(w // factor)

        # resize the image - down
        img = cv2.resize(img, (new_width, new_height), interpolation = cv2.INTER_LINEAR)

        # resize the image - up
        img = cv2.resize(img, (w, h), interpolation = cv2.INTER_LINEAR)

        # save the image
        print('Saving {}'.format(file))
        cv2.imwrite('/content/drive/MyDrive/imgg/{}'.format(file), img)
prepare_images('/content/drive/MyDrive/source/', 2)

#Testing Low Resolution Images

# test the generated images using the image quality metrics

for file in os.listdir('/content/drive/MyDrive/imgg/'):

```

```
# open target and reference images
target = cv2.imread('/content/drive/MyDrive/imgg/{}'.format(file))
```

```

ref = cv2.imread('/content/drive/MyDrive/source/{}'.format(file))

# calculate score
scores = compare_images(target, ref)

# print all three scores with new line characters (\n)
print('{}\nPSNR: {}\nMSE: {}\nSSIM: {}\n'.format(file, scores[0], scores[1], scores[2]))

#ssim - 0 complete diff match, 1 unique match
#psnr - higher = low error vice versa

#Building the SRCNN Model

# define the SRCNN model
def model():

    # define model type
    SRCNN = Sequential()

    # add model layers
    SRCNN.add(Conv2D(filters=128, kernel_size = (9, 9), kernel_initializer='glorot_uniform',
                    activation='relu', padding='valid', use_bias=True, input_shape=(None, None, 1)))
    SRCNN.add(Conv2D(filters=64, kernel_size = (3, 3), kernel_initializer='glorot_uniform',
                    activation='relu', padding='same', use_bias=True))
    SRCNN.add(Conv2D(filters=1, kernel_size = (5, 5), kernel_initializer='glorot_uniform',
                    activation='linear', padding='valid', use_bias=True))

    # define optimizer
    adam = Adam(lr=0.0003)

    # compile model
    SRCNN.compile(optimizer=adam, loss='mean_squared_error', metrics=['mean_squared_error'])
    return SRCNN

```

```
#Deploying the SRCNN
```

```
# define necessary image processing functions
```

```

def modcrop(img, scale):
    tmpsz = img.shape
    sz = tmpsz[0:2]
    sz = sz - np.mod(sz,
scale) img = img[0:sz[0],
1:sz[1]] return img

def shave(image, border):
    img = image[border: -border, border: -border]
    return img

# define main prediction function

def predict(image_path):

    # load the srcnn model with weights
    srcnn = model()
    srcnn.load_weights('/content/drive/MyDrive/3051crop_weight_200.h5')

    # load the degraded and reference images
    path, file = os.path.split(image_path)
    degraded = cv2.imread(image_path)
    ref = cv2.imread('/content/drive/MyDrive/source/{}'.format(file))

    # preprocess the image with modcrop
    ref = modcrop(ref, 3)
    degraded = modcrop(degraded, 3)

    # convert the image to YCrCb - (srcnn trained on Y channel)
    temp = cv2.cvtColor(degraded, cv2.COLOR_BGR2YCrCb)

    # create image slice and normalize
    Y = numpy.zeros((1, temp.shape[0], temp.shape[1], 1), dtype=float)
    Y[0, :, :, 0] = temp[:, :, 0].astype(float) / 255

    # perform super-resolution with srcnn
    pre = srcnn.predict(Y, batch_size=1)

    # post-process output
    pre *= 255
    pre[pre[:] > 255] = 255

```

```
pre[pre[:] < 0] = 0  
pre = pre.astype(np.uint8)
```

```

# copy Y channel back to image and convert to
BGR temp = shave(temp, 6)
temp[:, :, 0] = pre[0, :, :, 0]
output = cv2.cvtColor(temp, cv2.COLOR_YCrCb2BGR)

# remove border from reference and degraded
image ref = shave(ref.astype(np.uint8), 6)
degraded = shave(degraded.astype(np.uint8), 6)

# image quality calculations
scores = []
scores.append(compare_images(degraded,
ref)) scores.append(compare_images(output,
ref))

# return images and scores
return ref, degraded, output, scores

import numpy
ref, degraded, output, scores = predict('/content/drive/MyDrive/imgg/flo
wers.bmp')

# print all scores for all images
print('Degraded Image: \nPSNR: {}\nMSE: {}\nSSIM: {}\n'.format(scores[0]
[0], scores[0][1], scores[0][2]))
print('Reconstructed Image: \nPSNR: {}\nMSE: {}\nSSIM: {}\n'.format(scor
es[1][0], scores[1][1], scores[1][2]))

# display images as subplots
fig, axs = plt.subplots(1, 3, figsize=(20, 8))
axs[0].imshow(cv2.cvtColor(ref, cv2.COLOR_BGR2RGB))
axs[0].set_title('Original')
axs[1].imshow(cv2.cvtColor(degraded, cv2.COLOR_BGR2RGB))
axs[1].set_title('Degraded')
axs[2].imshow(cv2.cvtColor(output, cv2.COLOR_BGR2RGB))
axs[2].set_title('SRCNN')

# remove the x and y ticks
for ax in axs:
    ax.set_xticks([])
    ax.set_yticks([])

```

```
import numpy
```

```

ref, degraded, output, scores = predict('/content/drive/MyDrive/imgg/for
eman.bmp')

# print all scores for all images
print('Degraded Image: \nPSNR: {} \nMSE: {} \nSSIM: {} \n'.format(scores[0]
[0], scores[0][1], scores[0][2]))
print('Reconstructed Image: \nPSNR: {} \nMSE: {} \nSSIM: {} \n'.format(scores[1]
[0], scores[1][1], scores[1][2]))

# display images as subplots
fig, axs = plt.subplots(1, 3, figsize=(20, 8))
axs[0].imshow(cv2.cvtColor(ref, cv2.COLOR_BGR2RGB))
axs[0].set_title('Original')
axs[1].imshow(cv2.cvtColor(degraded, cv2.COLOR_BGR2RGB))
axs[1].set_title('Degraded')
axs[2].imshow(cv2.cvtColor(output, cv2.COLOR_BGR2RGB))
axs[2].set_title('SRCNN')

# remove the x and y ticks
for ax in axs:
    ax.set_xticks([])
    ax.set_yticks([])

for file in os.listdir('/content/drive/MyDrive/imgg'):

    # perform super-resolution
    ref, degraded, output, scores = predict('/content/drive/MyDrive/imgg
/{}'.format(file))

    # display images as subplots
    fig, axs = plt.subplots(1, 3, figsize=(20, 8))
    axs[0].imshow(cv2.cvtColor(ref, cv2.COLOR_BGR2RGB))
    axs[0].set_title('Original')
    axs[1].imshow(cv2.cvtColor(degraded, cv2.COLOR_BGR2RGB))
    axs[1].set_title('Degraded')
    axs[1].set_xlabel = 'PSNR: {} \nMSE: {} \nSSIM: {}'.format(scores[0][
0], scores[0][1], scores[0][2]))
    axs[2].imshow(cv2.cvtColor(output, cv2.COLOR_BGR2RGB))
    axs[2].set_title('SRCNN')
    axs[2].set_xlabel = 'PSNR: {} \nMSE: {} \nSSIM: {}'.format(scores[1]
[0], scores[1][1], scores[1][2]))

```

```
# remove the x and y ticks
for ax in axs:
```

```
ax.set_xticks([])
ax.set_yticks([])

print('Saving {}'.format(file))
fig.savefig('/content/drive/MyDrive/output/{}.png'.format(os.path.sp
litext(file)[0]))
plt.close()
```