ROBUST SINGLE-IMAGE SUPER-RESOLUTION VIA CNN'S AND TV-TV MINIMIZATION

PROJECT - II REPORT

Submitted by

RAMI REDDY GARI NAVISHNA (19RH1A04K0)
TENTU ALEKHYA (19RH1A04M7)
THADISHETTY AISHWARYA (19RH1A04M8)

Under the Esteemed Guidance of

Dr. Y. Madhaveelatha
(Professor of ECE & Principal)

in partial fulfillment of the Academic Requirements for the Degree of

BACHELOR OF TECHNOLOGY
Electronics and Communication Engineering

MALLA REDDY ENGINEERING COLLEGE FOR WOMEN
(Autonomous Institution-UGC, Govt. of India)
Accredited by NBA & NAAC with 'A' Grade
NIRF Indian Ranking, Accepted by MHRD, Govt. of India
Rank Band Excellent by ARIIA, Accepted by MHRD, Govt. of India
Maisammaguda, Dhulapally, Secunderabad – 500 100

2022-23

MALLA REDDY ENGINEERING COLLEGE FOR WOMEN

(Autonomous Institution-UGC, Govt. of India)

Accredited by NBA & NAAC with 'A' Grade
NIRF Indian Ranking, Accepted by MHRD, Govt. of India
Rank Band Excellent by ARIIA, Accepted by MHRD, Govt. of India
Maisammaguda, Dhulapally, Secunderabad – 500 100

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the Project - II work entitled "**ROBUST SINGLE-IMAGE SUPER-RESOLUTION VIA CNN'S AND TV-TV MINIMIZATION**" is carried out by **RAMI REDDY GARI NAVISHNA (19RH1A04K0), TENTU ALEKHYA (19RH1A04M7) & THADISHETTY AISHWARYA (19RH1A04M8)** in partial fulfillment for the award of degree of **BACHELOR OF TECHNOLOGY** in **ELECTRONICS AND COMMUNICATION ENGINEERING**, Jawaharlal Nehru Technological University, Hyderabad during the academic year 2022-2023.

Supervisor's Signature

Dr. Y. MADHAVEELATHA
Professor of ECE & Principal

Head of the Department

Dr. K. SUDHAKAR
Professor of ECE

External Examiner

Atal innovations
Learn. Innovate. Create.

PROJECT COMPLETION CERTIFICATE

This is to certify that the Following Student From **"MALLA REDDY ENGINEERING COLLEGE FOR WOMEN"** have Undergone Project Training Titled "**ROBUST SINGLE-IMAGE SUPER-RESOLUTION VIA CNN'S AND TV-TV MINIMIZATION**" in our Institute.

RAMI REDDY GARI NAVISHNA	**(19RH1A04K0)**
TENTU ALEKHYA	**(19RH1A04M7)**
THADISHETTY AISHWARYA	**(19RH1A04M8)**

They are successfully completed their project. This is based on bonafide work carried out electronics and communication division, ATAL INNOVATIONS under our guidance. Their performance and conduct during to this project was satisfactory.

For Atal Innovations

Sudaykiran

ATAL INNOVATIONS • HYDERABAD • Regd. No: 3012/2019

Authorized Signatory

ACKNOWLEDGEMENT

We feel ourselves honored and privileged to place our warm salutation to our college **Malla Reddy Engineering College for Women (Autonomous)** and Department of **Electronics and Communication Engineering** which gave us the opportunity to have expertise in engineering and profound technical knowledge.

We would like to deeply thank our Honorable Minister of Telangana State **Sri.Ch. Malla Reddy Garu,** founder chairman MRGI, the largest cluster of institutions in the state of Telangana for providing us with all the resources in the college to make our project success.

We wish to convey gratitude to our **Principal Dr. Y. Madhavee Latha,** for providing us with the environment and mean to enrich our skills and motivating us in our endeavor and helping us to realize our full potential.

We express our sincere gratitude to our **Head of the Department Dr. K. Sudhakar,** of for inspiring us to take up a project on this subject and successfully guiding us towards its completion.

We would also like to thank our Project coordinator **DR. S. A. Mary Rajee,** for his kind encouragement and overall guidance in viewing this program a good asset with profound gratitude.

We would like to thank our internal guide **Dr. Y. Madhavee Latha,** and all the Faculty members for their valuable guidance and encouragement towards the completion of our project work.

With Regards and Gratitude

RAMI REDDY GARI NAVISHNA (19RH1A04K0)
TENTU ALEKHYA (19RH1A04M7)
THADISHETTY AISHWARYA (19RH1A04M8)

DECLARATION

We hereby declare that our Project II entitled **"Robust Single-Image Super-Resolution via CNN's and TV-TV Minimization"** submitted to **Malla Reddy Engineering College for Women(Autonomous Institution),** affiliated to **Jawaharlal Nehru Technological University, Hyderabad** for the award of the Degree of **Bachelor of Technology** in **Electronics and Communication Engineering** is a result of original research work done by us.

It is declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of Degree.

RAMI REDDY GARI NAVISHNA (19RH1A04K0)
TENTU ALEKHYA (19RH1A04M7)
THADISHETTY AISHWARYA (19RH1A04M8)

ABSTRACT

Single-image super-resolution is the process of increasing the resolution of an image, obtaining a high-resolution (HR) image from a low-resolution (LR) one. By leveraging large training datasets, convolutional neural networks (CNNs) currently achieve the state-of-the-art performance in this task. Yet, during testing/deployment, they fail to enforce consistency between the HR and LR images: if we down sample the output HR image, it never matches its LR input. Based on this observation, we propose to post-process the CNN outputs with an optimization problem that we call *TV-TV minimization*, which enforces consistency. As our extensive experiments show, such post-processing not only improves the quality of the images, in terms of PSNR and SSIM, but also makes the super-resolution task robust to operator mismatch, i.e., when the true down-sampling operator is different from the one used to create the training dataset.

INDEX

INTRODUCTION

In Science and engineering, images acquired by sensing devices often have resolution well below the desired one. Common reasons include physical constraints, as in astronomy or biological microscopy, and cost, as in consumer electronics or medical imaging. Creating high-resolution (HR) images from low-resolution (LR) ones, a task known as *super- resolution* (SR), can therefore be extremely useful in these areas; it enables, for example, the identification of structures or objects that are barely visible in LR images. Doing so, however, requires inferring values for the unobserved pixels, which cannot be done without making assumptio[6]–[12]. More recently, data-driven methods have become very popular; their main assumption is that image features can be learned from training data, via dictionaries or via convolutional neural networks (CNN's).

CNNs were first applied to image SR in the seminal work [15] and have ever since remained the state-of-the-art, both in terms of reconstruction performance and computational complexity (during deployment/testing). By relying on vast databases of images for training, such as ImageNet [18] or T91 [13], they can effectively learn to map LR images/patches to HR images/patches. Although training a CNN can take several days, applying it to an image (what is typically called the testing phase) takes a few seconds or even sub-seconds. Despite these advantages, the knowledge that CNN's extract from data is never made explicit, making them hard to adapt to new scenarios: for example, simply changing the scaling factor or the sampling model, e.g., from bi-cubic to point sampling, almost always requires retraining the entire network. More conspicuously, however, is that during testing SR CNN's fail to guarantee the consistency between the reconstructed HR image and the input LR image, effectively ignoring precious “measurement” information, as we will illustrate shortly. Ignoring such information makes CNN's prone to generalization errors and, as a consequence, also less robust.

Curiously, adaptability and measurement consistency are the main features of classical reconstruction-based methods, which consist of algorithms designed to solve an optimization problem. To formulate such an optimization problem, one has to explicitly encode the measurement model and the assumptions about the class of images to be super-resolved.

Although this explicit encoding confers reconstruction-based methods great adaptability and flexibility, it naturally limits the complexity of the assumptions, which is one of the reasonswhy reconstruction-based methods are outperformed by data- driven methods

(CNN's). This motivates our problem.*Can we design SR algorithms that learn from large quantities of data and, at the same time, are easily adaptable to new scenarios and guarantee measurement consistency during the testing phase?* In other words, can we design algorithms that have the advantages of both data-driven and model-based methods?

A. Lack of Consistency by CNN's

∈ Before summarizing our method, we describe how CNN's fail to enforce consistency between the reconstructed HR image and the input LR image. Although we illustrate this phenomenon here for the specific SRCNN network [15], more systematic experiments can be found in Section IV. The top-left corner of Fig. 1 shows a ground truth (GT) image $X^\star \in \mathbb{R}^{M\times N}$, which the algorithms have access to during training, but not during testing.

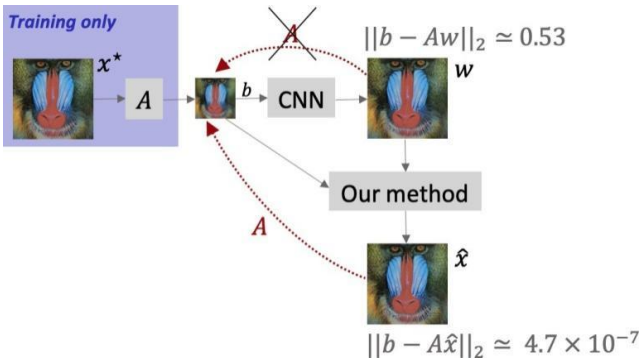


Fig. 1. Illustration of the lack of measurement consistency by CNN's during *testing*: when the output image w is down sampled using *A*, it typically differs significantly from *b*. Our method takes in both w and *b*, and fixes this problem.

We point out that *A* represents bi-cubic down sampling, which was what the authors of [15] assumed during the training of SRCNN. Although the data consistency (DC) problem has been given importance in other fields, only a few SISR techniques have tackled it. Existing supervised methods mainly rely on modifying the network structure [19], [20] or learn pretrained-denoisers and plug them in a model-based algorithm [21], [22].

1.1 What is Image Resolution?

Perhaps the most important technical concept to understand in imaging literature is the word resolution. Resolution is a fundamental issue in judging the quality of various image acquisition or processing systems. In its simplest form, *image resolution* is defined as the smallest discernible or measurable detail in a visual presentation. In optics the resolution of a device is determined by measuring the modulation transfer function (MTF) or the optical transfer

function (OTF) which represents the response of the system to different spatial frequencies. MTF is not only used to give the resolution limit at a single point, but also to characterize the system to an arbitrary input.

Researchers in digital image processing and computer vision classify resolution into three different types.

• Spatial Resolution: An image is made up of small picture elements called pixels. Spatial resolution refers to the spacing of the pixels in an image and is measured in pixels per unit length. The higher the spatial resolution, the more are the pixels in an image. High spatial resolution allows a clear perception of sharp details and subtle color transitions in an image. In case an image with high levels of details is not represented by a spatially dense set of pixels, the image is said to suffer from aliasing artifacts. For an output device such as a printer the spatial resolution is expressed in dots per inch (dpi).

• Brightness Resolution: Also known as gray-level resolution, it refers to the number of brightness levels or gray-levels used to represent a pixel. The brightness resolution increases with the number of quantization levels used. A monochrome image is usually quantized using 256 levels with each level represented by 8 bits. For a color image, at least 24 bits are used to represent one brightness level, i.e., 8 bits per color plane (red, green, blue). It should be noted that the number of gray value quantization levels is also intrinsically related to the spatial sampling rate. If the camera sensor has fewer quantization levels, it should have a much higher spatial sampling rate to capture the scene intensity. This idea is quite similar to that of delta modulation used in communication systems and to that of dithering used in half-tone printing.

• Temporal Resolution: It represents the frame rate or the number of frames captured per second. Higher the temporal resolution, lesser is the flicker observed. The lower limit on the temporal resolution is proportional to the amount of motion occurred between two consecutive frames. The typical frame rate for a pleasing view is about 25 frames per second or above.

Another kind of resolution of interest is the spectral resolution and it refers to the frequency or spectral resolving power of a sensor that gives the bandwidth of the light (or electro-magnetic wave) frequencies captured by the sensor. It is defined as the smallest resolvable wavelength difference by the sensor. The spectral resolution plays an important role in satellite imaging.
In this monograph, the term resolution unequivocally refers to the spatial resolution,

enhancement of which is the subject matter of this book. Further, we do not explore the inter-relationship between the brightness and spatial resolutions in this monograph.

1.2 Need for Resolution Enhancement

An image sensor or camera is a device which converts optical energy into an electrical signal. Modern imaging sensors are based on the charge-coupled device (CCD) technology, which consists of an array of photo-detector elements or pixels that have a voltage output proportional to the incident light. The number of detector elements decide the spatial resolution of the camera. Higher the number of detector elements, more is the resolution. A sensor with less number of detector elements produces a low resolution image, giving blocky effect. This is because when a scene is photographed with a low resolution camera, it is sampled at a low spatial sampling frequency, causing aliasing effect. One could think of reducing the size of the photo-detector elements, thereby increasing the density and hence the sampling rate. But as the pixel size decreases the amount of light incident on each pixel also decreases and this causes a shot noise, which degrades the image quality. Increasing the pixel density increases the resolution but also causes shot noise. The current image sensor technology has almost reached this level. Another approach to increase the resolution is to increase the wafer size which leads to an increase in the capacitance. This approach is not effective since an increase in capacitance causes a decrease in charge transfer rate. This limitation causes the image of a point light source to be blurred. Also there is distortion due to aliasing because of a low sampling rate for a low resolution sensor. Moreover in some applications like satellite imagery, the physical constraints make the sensor unrealizable for a high resolution. Thus there is a need for developing post acquisition signal processing techniques to enhance the resolution. These techniques being post processing methods applied on the low resolution images, they offer flexibility as well as cost benefit since there is no additional hardware cost involved. However, the increased computational cost may be the burden that an user has to bear.

1.3 Super-Resolution Concept

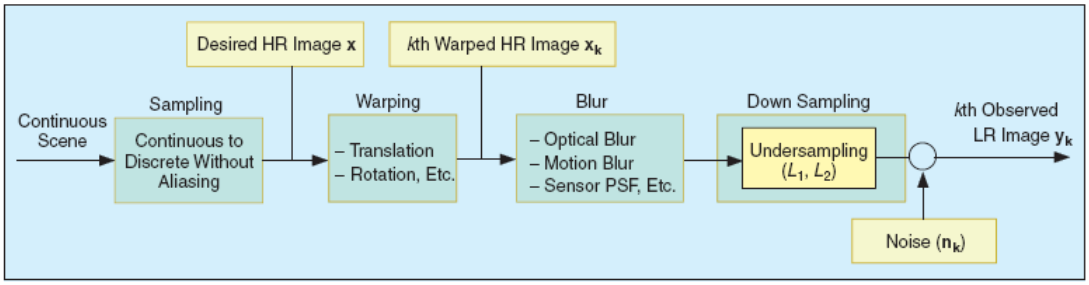
The low resolution representation resulting from the lower spatial sampling frequency produces distortion in the image due the to loss of high frequency components. This causes loss of important information such as edges and textures. Also a degradation occurs due to the sensor point spread function (PSF), and optical blurring due to camera motion or out-of-focus. Thus an image captured with a low resolution camera suffers from aliasing, blurring and presence of noise. *Super-resolution* (SR) refers to the process of producing a high spatial resolution image

from several low resolution images, thereby increasing the maximum spatial frequency and

removing the degradation's that arise during the image capturing process using a low resolution camera. In effect, the super-resolution process extrapolates the high frequency components and minimizes aliasing and blurring. As already mentioned, one way to increase the sampling rate is to reduce the pixel size, thereby increasing the pixel density. But an in crease in pixel density causes shot noise and hence the distortion. Also the cost of sensor increases with the increase in pixel density. Hence the sensor modification is not always a practical solution for increasing the resolution. Thus we resort to image processing techniques to enhance the resolution. The advantage here is that there is no additional hardware cost involved and also it offers a flexibility such as region of interest super-resolution. One of the approaches towards this end is simple image interpolation that can be used to increase the size of the image. But the quality of the interpolated image is very much limited due to the use of a single, aliased low resolution image. Also the single image interpolation is highly ill-posed problem since there may exist infinitely many upsampled or expanded images which are consistent with the original data. A single image interpolation cannot recover the high frequency components lost or degraded due to the low resolution sampling. Some progress can be achieved by convolving the image with a filter designed to boost the higher frequency components. Unfortunately this also amplifies any noise in the image and degrades the quality. Hence the image interpolation methods are not considered as super-resolution techniques. In order to obtain super-resolution we must look for nonredundant information among the various frames in an image sequence. The most obvious method for this seems to be to capture multiple low resolution observations of the same scene through subpixel shifts due to the camera motion. These subpixel shifts can occur due to the controlled motion in imaging systems, e.g., a landsat satellite captures images of the same area on the earth every eighteen days as it orbits around it. The same is true for uncontrolled motion, *e.g.,* movements of local objects or vibrating imaging systems. If the low resolution image shifts are integer units, then there is no additional information available from subsequent low resolution observations for super-resolution purposes. However, if they have subpixel shifts then each low resolution aliased frame contains additional information that can be used for high resolution reconstruction. Such a technique makes the ill-posed nature of the problem to a better-posed one, as more data is available from multiple frames. Many researchers often term the process of super-resolving a scene as super-resolution restoration. It may be mentioned here that super resolution differs from a typical image restoration problem wherein the image formation model (discussed in the next section) does not consider the decimation process i.e., the aliasing which is inherently present in the low resolution observations. Thus the size of the restored image is the same as that of the observed image for image restoration while it is dependent on the decimation factor for a super-resolved image.

1.4 Super-Resolution Technique

The success of any super-resolution reconstruction method is based on the correctness of the low resolution image formation model that relates the original high resolution image to the observed images. The most common model used is based on observations which are shifted, blurred and decimated (aliased) versions of the high resolution image. The observation model relating a high resolution image to low resolution video frames is shown in Figure 1.3.1.

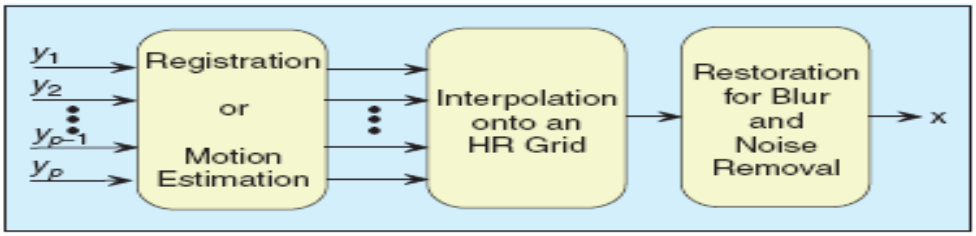


Observation model relating LR images to HR images

Let us assume that the scene is static and the camera is slowly moving. Let us further assume that the depth variation in the scene is negligible compared to its distance from the camera so that the perspective distortion due to camera motion can be neglected. In the figure *z(x,y)* is the desired high resolution image which is obtained by sampling the spatially continuous scene at a rate greater than or equal to the Nyquist rate. Here the assumption is that the continuous scene is bandlimited. The camera motion at the *kth* time instant during the exposure is modeled as pure rotation θk and translation *tk*. Next, the blurring which may be caused by the optical system or due to relative motion between the camera and the scene, can be modeled as linear space invariant or linear space variant. One can select an appropriate point spread function (PSF) for the blur. These warped and blurred high resolution images undergo a low resolution scanning, i.e., sub sampling or decimation, followed by noise addition, yielding the low resolution observations. Most of the super-resolution methods proposed in the literature use motion between the observed frames as a cue for estimating the high resolution image. This being the most intuitive approach for super resolution, is based on a three-stage algorithm consisting of registration, interpolation and restoration. The registration step is used to find the relative motion between the frames with a subpixel accuracy. The assumption here is that all the pixels from the available frames can be mapped back onto the reference frame based on the motion vector information. Or in other words, there is no occlusion which is usually true if the

depth variation on the scene is planer. Next, the interpolation onto a uniform grid is done to obtain a uniformly spaced upsampled image.

Once the upsampled image on uniformly spaced grid points is obtained, restoration is applied to remove the effects of aliasing and blurring and to reduce noise. The restoration can be performed by using any deconvolution algorithm that considers the presence of an additive noise. A scheme for constructing the high resolution frame from multiple low resolution frames is shown in Figure 1.3.2.

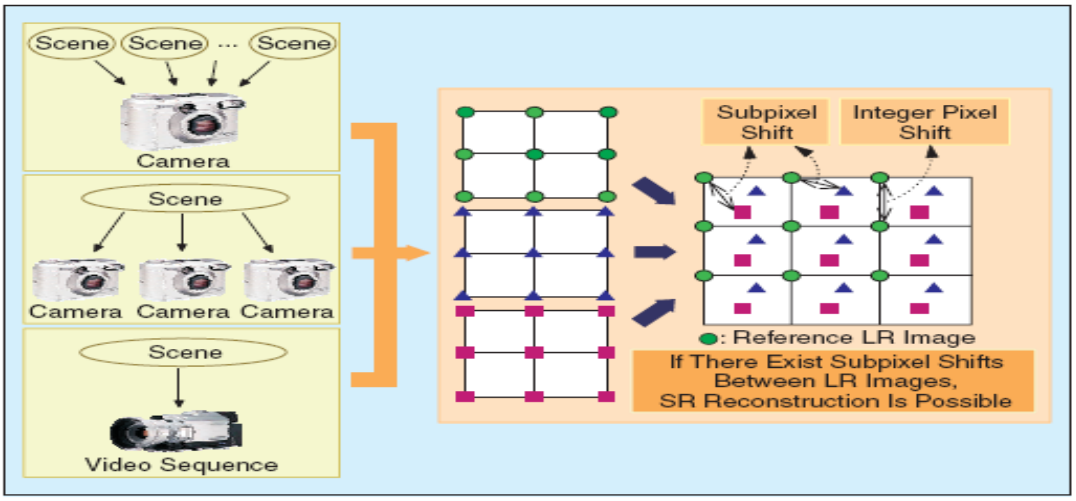


Scheme for super-resolution from multiple subpixel shifted observations.

Here the low resolution observations y1, y2,…,yp are used as input to the motion estimation module. The registered images are then interpolated onto a high resolution grid, which is then post-processed through restoration to generate a super-resolved image.

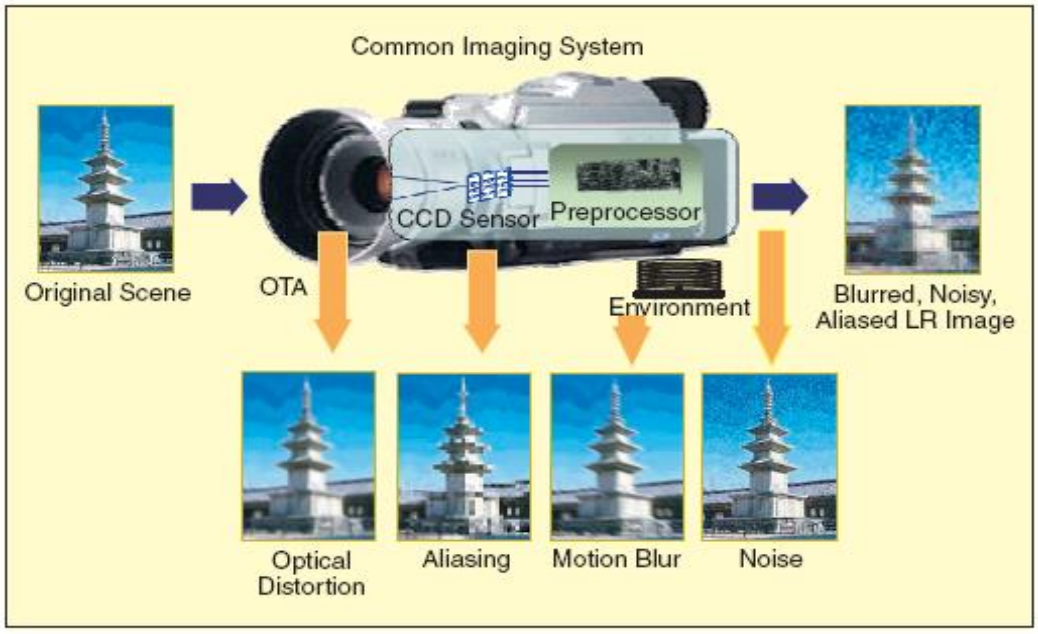
1.5 The idea to get HR image from multiple LR image

The basic premise for increasing the spatial resolution in SR techniques is the availability of multiple LR images captured from the same scene. In SR, typically, the LR images represent different “looks” at the same scene. That is, LR images are subsampled (aliased) as well as shifted with subpixel precision. If the LR images are shifted by integer units, then each image contains the same information, and thus there is no new information that can be used to reconstruct an HR image. If the LR images have different subpixel shifts from each other and if aliasing is present, however, then each image cannot be obtained from the others. In this case, the new information contained in each LR image can be exploited to obtain an HR image. To obtain different looks at the same scene, some relative scene motions must exist from frame to frame via multiple scenes or video sequences. Multiple scenes can be obtained from one camera with several captures or from multiple cameras located in different positions. These scene motions can occur due to the controlled motions in imaging systems, e.g., images acquired from orbiting satellites. The same is true of uncontrolled motions, e.g., movement of local objects or vibrating imaging systems. If these scene motions are known or can be estimated within subpixel accuracy and if we combine these LR images, SR image reconstruction is possible as illustrated in Figure 1.4.1.



Basic premise for super resolution

In the process of recording a digital image, there is a natural loss of spatial resolution caused by the optical distortions (out of focus, diffraction limit, etc.), motion blur due to limited shutter speed, noise that occurs within the sensor or during transmission, and insufficient sensor density as shown in Figure 1.4.2.



Common image acquisition system

Thus, the recorded image usually suffers from blur, noise, and aliasing effects. Although the main concern of an SR algorithm is to reconstruct HR images from undersampled LR images, it covers image restoration techniques that produce high quality images from noisy, blurred images. Therefore, the goal of SR techniques is to restore an HRimage from several degraded and aliased LR images. The goal of image restoration is to recover a degraded (e.g., blurred, noisy) image, but it does not change the size of image.

In fact, restoration and SR reconstruction are closely related theoretically, and SR reconstruction can be considered as a second-generation problem of image restoration. Another problem related to SR reconstruction is image interpolation that has been used to increase the size of a single image. Although this field has been extensively studied, the quality of an image magnified from an aliased LR image is inherently limited even though the ideal sinc basis function is employed. That is, single image interpolation cannot recover the high-frequency components lost or degraded during the LR sampling process. For this reason, image interpolation methods are not considered as SR techniques. To achieve further improvements in this field, the next step requires the utilization of multiple data sets in which additional data constraints from several observations of the same scene can be used. The fusion of information from various observations of the same scene allows us SR reconstruction of the scene.

Our Approach

Our algorithm can be viewed as a post-processing step that takes as input the CNN image w and the LR image b, and reconstructs a HR image $\hat{x} \in \mathbb{R}^n$ that is similar to w but, in contrast to it, satisfies $Ax = b$ (bottom-right corner of Fig. 1). As a result, the images created by our method almost always have better quality than w, in terms of PSNR and SSIM. In addition, our method confers robustness to the SR task, even when the operator A used to generate the training data differs from the one used during testing. We integrate b and w via an algorithm that solves an optimization problem that we call *TV-TV minimization*. The problem enforces the reconstructed image to have a small number of edges, a property captured by a small TV-norm, and also to not differ much from w, as measured again by the TV-norm. Naturally, it also imposes the constraint $A\hat{x} = b$.

Contributions

We summarise our contributions as follows:

1) We introduce a framework that has the advantages of learning-based and reconstruction-based methods. Like reconstruction methods, it is adaptable, flexible, and enforces measurement consistency. At the same time, it retains the excellent performance of learning methods.
2) We integrate learning and reconstruction-based methods via a TV-TV minimization problem. Although we have no specific theoretical guarantees for it, existing theory for a related, simpler problem (4_1-4_1 minimization) pro- vides useful insights about how to tune a regularization parameter.

3) We propose an algorithm based on the alternating direc- tion method of multipliers (ADMM) [23] to solve the TV-TV minimization problem. In contrast with most SR methods, which process image patches independently, our algorithm processes full images at once. It also easilyadapts to different degradations and scaling factors.
4) We conduct extensive experiments that illustrate not only the robustness of our algorithm under different degradation operators, but also how it systematically improves (in terms of PSNR and SSIM) the outputs of state-of-the-art SR networks, such as EDSR [24] and RCAN [25]

We highlight the following differences with respect to our previous work in [1], [2]. We now explore and illustrate with experiments the underlying reason why our framework improves the output of state-of-the-art SR CNNs. We also describe how the proposed optimization problem can be solved efficiently using ADMM; in fact, the algorithms used in [1], [2] were different and less efficient than the algorithm we present here. Our experiments are also much more extensive: they consider different sampling operators to illustrate robust- ness to operator mismatch, and include many more algorithms,

LITERATURE REVIEW

Single-image super-resolution (SR) aims at restoring a high resolution (HR) image from a single low-resolution (LR) input image. Since multiple HR images can be recovered from a single LR observation, single image SR is heavily ill-posed. To address this problem, additional prior information is required to constrain the solution space. Internal statistics of a single natural image provides useful priors [1], and has shown strong power to solve the SR problem [2, 3, 4, 5, 6, 7, 8].

Patches in a natural image tend to redundantly recur many times inside the image, both within the same scale, as well as across different scales [2]. Based on this internal statistical prior, Glasner et al. [2] presented a unified SR framework that combines classical multi-image and example-based SR methods. Yang et al. [3] proposed a SR method that exploits self-similarities and group structural information of image patches using only a single input image. Freedman and Fattal [4] followed a local self-similarity assumption on natural images and extracted nearest neighbours from extremely localized regions in the input image, accordingly reducing considerably the nearest-patch search time. Similar to [2], Zhang et al. [5] exploited similarity redundancy across different scales in a given LR image itself to achieve example-based SR using the neighbour embedding algorithm [9], and applied the nonlocal means method [10] to learn the similarity within the same scale. Bevilacqua et al. [6] made use of a ”double pyramid” of images, built starting from the input image itself, to extract the self-examples, and then employed a regression based method to directly map each LR patch into its HR version. To expand the internal patch space and achieve more stable nearest-patch search, Choi $x_h = P_k x_l$ et al. [7] utilized context dependent multi-shaped sub patches and Huang et al. [8] used a factored patch transformation representation for simultaneously accounting for both planar perspective distortion and affine shape deformation of image patches.

Apart from the above internal example-based approaches, another kind of example-based method uses an external database of natural images to extract a large number of training LR and HR image patch pairs. These methods usually use machine learning techniques to learn the relationship between LR and HR image patches. Timofte et al. proposed an anchored neighbourhood regression method [11] and its refined variant called A+ [12], where learned dictionary elements are used as the anchor points and multiple linear regressors are computed to map LR to HR feature subspace. Recently, deep learning was introduced to address the single image SR problem. Dong et al. [13] proposed a Super-Resolution Convolutional Neural Network (SRCNN) model that directly learns an end-to-end mapping between LR and HR images.

Literature survey table on Super-Resolution Techniques

S.No.	Title of the paper	Name of the Journal	Year	Method	Probable enhancement	Metric	Advantages	Disadvantages
1	Image Super-Resolution Using Deep Convolutional Networks.	IEEE	2015	SRCNN	Visual quality improvement	PSNR (44.35)	Simplicity and robustness and can be applied to other low level vision problems	PSNR values of CB, CR channels are poorer than Bicubic interpolation it is useful to give better for CE single channel network.
2	Deep Networks for Image Super-Resolution with Sparse Prior	IEEE	2015	CSCN	Visual quality improvement	PSNR (40.15)	Both quantitative and qualitative performance than deep and shallow SR models. Minimizes reconstruction errors increasing model parameters and training data.	Ringing artifacts and blurs on corners and fine structures due to the lack of self patches.
3	Robust Single Image Super-Resolution via Deep Networks With Sparse Prior	IEEE	2016	CSCN	Visual quality improvement	PSNR (37.00)	Enhanced training speed and model compactness. Fixed and incremental scaling factors to enhance SR performance.	Many artifacts induced by noise are observed by SR result.
4	Anchored Neighborhood Regression for Fast Example Based Super-Resolution	IEEE	2013	ANR	Visual quality improvement and fast execution time	PSNR (35.83) Execution time(0.78)	Better speed performance, fast execution while retaining the quality performanc	High execution speed in exchange for some visual quality loss.

							e.	
5	A+: Adjusted Anchored Neighborhood Regression for Fast Super-Resolution	Asian Conference on Computer Vision (ACCV)	2014	A+	Visual quality improvement and fast execution time	PSNR (36.55) Execution time(0.55)	Lowest time complexity and substantially better performance.	Not suitable for real time critical applications, better performance at price of increasing the training data.
6	Single Image Super-Resolution from Transformed Self-Exemplars	IEEE	2015	SR ALGORITHMS	Visual quality improvement	PSNR (31.12) SSIM (0.88)	Out performance than existing SR algorithms without using external training samples. Improvisation of the self-exampler search with 3D scene geometry and patch search	The planes are not accurately detected, fail to recover the regular structure, more processing time
7	Anchored Neighborhood Regression Based Single Image Super-Resolution From Self Examples	IEEE	2016	SR ALGORITHMS	Visual quality improvement	PSNR (36.89) SSIM (0.9629)	Outperformance than single image SR algorithm both objective and subjective quality. It results with fever artifacts, sharper edges and finer details.	This method may not satisfy the reconstruction constrain exactly. It is a time consuming to train fast dictionary.
8	Single Image Super-Resolution via Locally Regularized Anchored Neighborhood	IEEE	2016	LANR-NLM	Visual quality improvement	PSNR (31.93) SSIM (0.8958)		

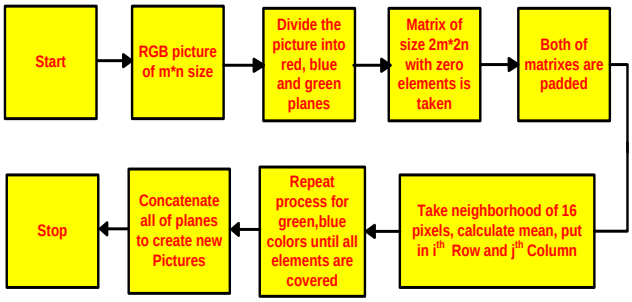
	Regression and Nonlocal Means							
9	Learning-Based Joint Super-Resolution and Deblocking for a Highly Compressed Image	IEEE	2015	Praposed Sparse coding super resolution (SCSR)	fast execution time	Execution time (121.9 s)		
10	Learning-Based joint Super-Resolution and Deblocking for a Highly Compressed Image	IEEE	2015	Sparse representation and MCA based image decomposition	Visual quality improvement	Time (153.74)		
11	Bayesian Sparse Representation for Hyper spectral Image Super Resolution.	IEEE	2015	Bayesian sparse coding method	Visual quality improvement	Time(180)		
12	Fusion of hyperspectral and multispectral images using spectral unmixing and sparse coding.	Research Gate	2016	SUSC	Visual quality improvement and fast execution time	PSNR (32.3) TIME (551.36)		
13	Hyperspectral Image resolution enhancement based on joint sparsity spectral unmixing.	Research Gate	2014	MLJSF	Visual quality improvement	nRMSE (3.2%) NCC(0.98)		

14	Hyper spectral image super-resolution via non-negative structured sparse representation	IEEE	2016	NSSR	fast execution time	PSNR (42.26) RMSE (2.21) SAM(4.33) ERGAS (0.30)		

EXISTING METHOD

Many interpolation techniques have been developed to increase the image resolution. The three well-known linear interpolation techniques are nearest neighbor interpolation, bilinear interpolation, and bi-cubic interpolation. Bi-cubic interpolation is more sophisticated than the other two techniques, it produces noticeably sharper images.

3.1 BICUBIC INTERPOLATION: It is better than nearest neighbor and bilinear interpolation methods. It is produced the smoother picture. Picture produces by it is near to the original picture. As compare to nearest neighbor, neighborhood of 16 pixels is used in bicubic interpolation method. For scaling pictures, bicubic interpolation is used. In bicubic interpolation, fine detail of picture is preserved. Processing time is more and picture quality is better in bicubic interpolation [3]. Flowchart of bicubic interpolation method is as follow:



Flowchart of Bicubic interpolation

Super-resolution (SR) aims to produce an high resolution (HR) image, given one or more low resolution (LR) images as input. In contrast to simple interpolation techniques like the ones described in Section 2.1, SR methods aim at recovering or estimating the information missing from the low-resolution image. There are multiple ways to approach the problem and SR algorithms can be classified in numerous ways. The simplest way is to classify them either single image SR (SISR) or multi-image SR (MISR), based on the number of low resolution input images used. MISR tackles the problem with multiple different images depicting the same scene, with each image having different sub-pixel alignments (translation, rotation etc.), different scales, different blurring, or other similar variations between them. As a single low resolution image could have been produced by scaling down infinite number of different high resolution images, using multiple different images introduces additional constraints for the possible high resolution source and thus enables the reconstruction of the high resolution details.

In many practical applications only a single low resolution image is available, which is why SISR algorithms have attracted more research interest in recent years. SISR algorithms can be coarsely split into two different groups. Methods of the first group use only the information available in the source image by exploiting the self-similarity of the scene, whereas the second group utilizes external databases of different LR–HR image pairs. Self-similarity based methods split the image into smaller patches and super-resolve those patches individually using similar patches found elsewhere from the image. Ideally those patches would represent the identical objects, but with different scales, translations, rotations etc. like in the case of MISR. For example, many typical man made scenes
i.e. urban environments contain a lot of recurring elements which can be utilized in super-resolution.

Algorithms using external image databases employ machine learning techniques to find a typical mapping from high to low resolution, and use that knowledge to estimate a probable high resolution image from the low resolution input. These techniques work especially well in applications where the image content is limited to specific cases i.e. human faces or hand written characters. Multiple different machine learning techniques can be used but especially convolutional neural networks and other deep learning methods have proven popular in recent years.

This chapter will give an overview of different approaches to super-resolution in a form of literature review. The next two sections will give overviews of MISR (Section 3.1) and SISR (Section 3.2) in general. Section 3.3 will briefly describe SISR algorithms based on self-similarity. The main focus of this thesis is on methods based on convolutional neural networks, and Section 3.5 will describe those. Methods based on more classical machine learning techniques will be discussed briefly in Section 3.4.

3.2 Multi-image super-resolution

All digital imaging systems, whether a digital still camera or i.e. a flatbed scanner, produce a discrete, sampled version of the continuous objects present in the scene. As was described in the Section 2.1, frequencies in the original signal higher than half of the sampling frequency will be folded to lower frequencies causing aliasing artifacts. To prevent these artifacts, cameras employ an optical low-pass filter in front of the sensors that blurs the image before sampling. Ideal anti-aliasing filter is impossible to produce, and thus some aliasing will always occur. These aliased frequencies are indistinguishable in a single image, but when there are multiple

unique images depicting the same scene, those frequencies can be recovered. This phenomenon is exploited in classical MISR [17]. MISR algorithms were the first ones applied on digital images, but the concept of super- resolution imaging was introduced even before the prevalence of digital imaging. First papers concentrated on the so called optical SR, which tries to overcome the optical resolution limit caused by diffraction, instead of digital sampling [41, 65]. The first optical SR algorithm was introduced by Gerchberg [16] in 1974, although the theoretical foundation was established ten years earlier by Harris.

The first application of super-resolution on digital images was by Tsai and Huang [59] in 1984, which used multiple low resolution images to produce a single high resolution image. The method by Tsai and Huang, like the optical SR methods before it, worked in frequency domain. However, frequency domain SR has its limitations [65], and due to that spatial domain methods are more prevalent in modern research [41, 65].

The first spatial domain method was introduced by Peleg et al. [43], and it was based on sub-pixel displacements between the images. Irani and Peleg improved the spatial domain MISR further in 1991 [27] by introducing the iterative back-projection algorithm. This algorithm works by first producing an initial estimate of the HR image, and then producing simulated low-resolution images from the initial estimate. The simulated low-resolution images are compared to the original input images, and if the HR estimate is correct, the images should be identical. If there are differences between the images, the error is back-projected to the high resolution estimate to produce a new estimate and this process is iteratively repeated until the error is minimized. The iterative back-projection has been popular in many subsequent works, including SISR algorithms [11, 26].

There are also other approaches to multi-image super-resolution, but they are out of the scope of this thesis. All MISR algorithms are typically reconstruction based algorithms, which recover only information already existing but indistinguishable in the low resolution images. For these algorithms to work, it is required that every input image contains unique information. As the low resolution images have limited information, the performance of these methods is also limited especially in high scaling factors. To alleviate this problem, the so called hallucination based methods have been introduced. They can utilize information gathered from external database of HR–LR correspondences and create details that do not exist in the low resolution image. Although this approach could be integrated to multi-image reconstruction methods, it is mostly employed in single image SR and thus will discussed in more detail in the next section [41].

3.3 Single image super-resolution

In many applications it is infeasible or even impossible to acquire multiple images of the same scene, which has lead to development of SISR algorithms [41]. This approach has been the

main interest in recent SR research, mostly because of its broad applicability but also due to recent advances in machine learning techniques which have improved the performance significantly [6, 30, 67].

To recreate the information missing in the low resolution, extra information has to be acquired somehow. Since only a single image is available, the reconstruction approach of MISR is impossible. The details have to be created by either using machine learning methods with external database of LR–HR exemplar pairs, or utilizing the self-similarity of the low resolution image. The latter method is enabled by the fractal nature of typical scenes, as image elements tend to recur throughout the image in different scales, translations, rotations etc. [26]. Self-similarity methods can utilize some of the reconstruction techniques typical of MISR algorithms [17], as the image is processed in patches and similar patches can be considered as individual images depicting the same object.

Self-similarity based methods have the advantage of not requiring an extensive database of training images, but they have other drawbacks which have shifted the research focus towards learning-based methods. Self-similarity based methods have typically high computational cost for the processing of an image, since an extensive search of similar patches is required for each image patch processed [26]. Learning-based methods require a computationally intensive training process. However it is done only once and the super-resolving of a single image is relatively fast in comparison to self-similarity based methods. Self-similarity based methods produce good results only for images that have large amounts of similar patches, which is not true for most natural images [26]. In those cases machine learning methods typically outperform those based on self-similarity. Popularity of learning-based methods has also increased due to recent advances in convolutional neural networks and other deep learning techniques utilized in other image processing applications and they have large potential also in super-resolution [6, 30, 67].

3.4 Self-similarity based SISR

First single image SR algorithm utilizing only the self-similarity within a low resolution image was introduced by Ebrahimi and Vrscay in 2007 [9]. It was based on local scale invariance, which means that image patches are similar to themselves within small scaling factors. For example, a sharp and straight transition between two flat surfaces will appear identical across different scales. Glasner et al. took a different approach in 2009 [17], with an algorithm based on the observation that typically similar image patches occur repeatedly within an image, both with the same scale and different scales.

The low resolution image is processed patch by patch and for each patch the image is searched for one or more similar patches. If multiple patches within same scale are found, those can be used with traditional multi-image SR methods for scaling up the input patch. If at least one similar patch with larger scale is found, it can be used directly for estimating the missing high-frequency contents.

Freedman and Fattal combined the two approaches mentioned above and improved on the computational costs, by searching for similar patches within the local neighborhood of the target patch [13]. They also applied the method for video in addition to still images. Until 2015 all of the self-similarity based methods had considered only translated, scaled and rotated patches when searching similar patches, but Huang et al. [26] introduced SelfExSR, that utilizes also perspective and affine transformations. Their method was aimed specifically at images of buildings, urban environments and other man-built structures. Those images typically contain planar structures with recurring elements, whose perspective distortion can be easily estimated. Their method performs comparably to other methods of the time in terms of PSNR, but the computational complexity of SelfExSR is significantly higher than of the competing methods.

Self-similarity based methods have been used extensively also in other image processing tasks, especially in denoising, and some of those methods have inspired also SISR algorithms. A notable example of this is the BM3D algorithm by Dabov et. al. [5] originally published in 2007. This BM3D paradigm has been applied for super-resolution by Egiazarian et. al. first in 2007 [10] and later on in 2015 [11]. It utilizes both sparsity and non-local self-similarity by searching for a group of similar patches, which are arranged to a three-dimensional block.

3.5 Reconstruction-Based SR

Reconstruction-based schemes view SR as an image recon- struction problem and address it by formulating an opti- mization problem. In general, the optimization problem has two terms: a DC term that encodes assumptions about the acquisition process, usually that $Ax\ b$ (where x is the optimization variable), and a regularization term on x that encodes assumptions about the class of images. Different methods differ mostly on the image assumptions.

1) *Image Assumptions:* Reconstruction SR methods encode assumptions about the images by penalizing in the opti- mization problem measures of complexity. These reflect the empirical observation that natural images have parsimonious representations in several domains. Examples include sparsity in the wavelet domain [29], sparsity of image patches in the DCT domain [30] and, as we will explore shortly in more detail, sparsity of image gradients [6]–[12]. Since sparsity is well captured by the 4_1-norm, the resulting optimization problem is typically convex and can be solved efficiently. A more challenging assumption is multi-scale recurrence [24], [31], which captures the notion that patches of natural images occur repeatedly across the image.

2) *Total Variation:* In natural images, the number of pixels that correspond to an edge, i.e., a transition between different objects, is a small percentage of the total number of pixels. This can be measured by the total variation (TV) of the image [6]. Although TV was initially defined in the context of partial differential equations, there has been work that discretizes the differential equations [9], [10] or that directly defines TV in the discrete setting [11], [32]. Although there are several definitions of discrete TV, the most popular are the *isotropic TV,* which consists of the sum (over all pixels) of the 4_2-norms of the vectors containing the horizontal and vertical differences at each pixel, and the *anisotropic TV,* which is similar to isotropic TV but with the 4_2-norms replaced by the 4_1-norm. Both definitions yield convex, yet nondifferentiable, functions. Many algorithms have been proposed to solve prob- lems involving discrete TV, including primal-dual methods [8], [33], and proximal and gradient-based schemes [11], [12].

The concept of TV has been used in many imaging tasks, from denoising [6], [11] to SR [9], [10]. For example, [9] discretizes a differential equation relating variations in the values of pixels to the curvature of level sets, while enforcing fidelity to the LR image.

3) Back-Projection: Back-projection (BP) is an iterative algorithm originally proposed for multi-image SR [34]. It minimizes the reconstruction error and then projects the outputs back to the GT image to adjust its intensity. Although this improves the outputs, it is prone to ringing and chessboard artifacts.

3.6 Learning-Based Algorithms

Learning-based algorithms typically consist of two stages: *training*, in which a map from LR to HR patches is learned from a database of training images, and *testing*, in which the learned map is applied to super-resolve an unseen image.

1) *Dictionary Learning:* In dictionary learning, also known as sparse coding, patches of HR images are assumed to have a sparse representation on an over-complete dictionary, which is learned from training images. For example, [13] uses training images to learn dictionaries for LR and HR patches while constraining corresponding patches to have the same coefficients. Other schemes use similar concepts, but require no training data at all. For example, [14] uses self-similarity to learn the LR-HR map without any external database of images.

2) *CNN-Based Methods:* The advent of deep learning and the availability of large image datasets inspired the application of CNNs to SR. Currently, they surpass any reconstruction- or interpolation-based method both in reconstruction perfor- mance and in execution time (during testing). The first CNN for SR was proposed by [15]; although its design was inspired by dictionary learning methods, the proposed architecture set a new standard for SR performance.

SR networks can be classified as direct or progressive. In direct networks, the LR image is first upscaled, typically via bicubic interpolation, to the required spatial resolution, and then is fed to a CNN, as in [21], [24], [25]. In this case, the CNN thus learns how to deblur the upscaled image. As pre- viously mentioned, CNN architectures need to be retrained every time we change the scaling factor. To overcome this, [35] repeatedly applied a recursive convolutional layer to obtain the super-resolved image. However, since the LR input is blurry, the CNN outputs a HR image lacking fine details. Inspired by this observation, [36] proposed the SRGAN, which produces photo-realistic HR images, even though they do not yield the best PSNR. As direct networks operate on high-dimensional images, their training is computationally expensive [37].

Progressive networks, in contrast, have reduced training complexity, as they directly process LR images. Specifically, the upsampling step, performed using sub-pixel or transposed convolution [37], is applied at the end of the network. For instance, LapSRN [27] used the concept of Laplacian pyramids, in which each network level is trained to predict residuals between the upscaled images at different levels in the pyramid.

Most CNNs are trained with images that have been downsampled with a bicubic filter. Thus, their performance degrades when the true downsampling operator is different. Indeed, during testing, the true degradation is unknown.

3.7 Plug-and-Play Methods

A different line or work blends learning- and model- based methods. The main observation is that, when solving linear inverse problems, proximal-based algorithms separate the operations of measurement consistency and problem regu- larization (using prior knowledge). The latter usually consists of a simple operation, like soft-thresholding, which encodes the assumptions about the target image and which can be viewed as a denoising step. Given its independence from the measurements, such operation can be replaced by a more complex function, such as a CNN. The resulting algorithms are versatile, as the measurement operator can be easily modified. Most work in this area, however, has focused on compressed sensing, in which the measurement operator is typically a dense random matrix; see.

PROPOSED METHOD

4.1 PROPOSED FRAMEWORK

Main Model and Assumptions

We aim to reconstruct the vectorized version of an HR image $x^{\wedge} \in \mathrm{R}^n$ from a LR image $b \in \mathrm{R}^m$, with $m < n$. We assume that these quantities are linearly related: $b = Ax^{\wedge}$,

where $A \in \mathbb{R}^{m \times n}$ represents the down-sampling operator. The model in (1) is often used in reconstruction-based and dictionary learning algorithms [10], [13], [29], even though many methods also consider additive noise: $b = Ax^{*} + \epsilon$, , where ϵ is a Gaussian random vector More interesting, however, is that CNN-based methods implicitly assume the model in (1), although that is rarely acknowledged. In particular, all the SR networks we know of (e.g., [15], [16], [26], [27]) are trained with HR images that are downsampled according to (1), where A implements bicubic downsampling. We next discuss other possible choices for A.

- *Common Choices for A:* Different instances of $A \in \mathrm{R}^{m \times n}$ in (1) have been assumed in the SR literature:

· ***Simple Subsampling*:** A contains equispaced rows of the identity matrix $I_n \in \mathrm{R}^{n \times n}$, i.e., each row of A is a canonical vector (0, … , 0, 1, 0, …, 0). This operator is simple to implement, but often introduces aliasing.

· ***Bicubic*:** $A = S.B$, where S is a simple subsampling operator, and B is a bicubic filter. It is the operator of choice for processing training data for CNNs.

· ***Box-Averaging*:** if the scaling factor is s, then each row of A contains s^2 nonzero elements, equal to $1/s^2$, in positions corresponding to a neighborhood of a pixel. In other words, box-averaging replaces each block of $s \times s$ pixels by their average. Although simpler than the bicubic operator, it does not introduce the aliasing that simple subsampling does; see, e.g., [41].

In our experiments, we will mostly instantiate A as a bicubic operator. The reason is that most SR CNNs assume this oper- ator during training. Simple subsampling and box-averaging will be used to illustrate how our post-processing scheme adds robustness to operator mismatch, i.e., when A is different during training and testing.

Assumptions: We estimate $x^{\wedge} \in \mathrm{R}^n$ from $b \in \mathrm{R}^m$ bytaking into account two possibly conflicting

assumptions:

1) x^* has a small TV; a property that indicates the recon- structed image is close to the GT. For example, the aver-age TV norm of the BSD100 dataset is 0.08.

2) x^* is also close to the prior information w, the image returned by a learning-based method (CNN), where the notion of distance is also measured by TV.

4.2 Our Framework

The framework we propose is shown schematically in Fig. 2. It starts by super-resolving b into w Rn with a base method, which we assume is implemented by a CNN due to their current outstanding performance. As explained in Section I, CNNs fail to enforce measurement consistency during testing, i.e., Aw = b for any matrix A that is assumed to implement the downsampling operation. We propose to use an additional block that takes in both the HR output w of the CNN and the LR image b, and creates another HR image x. The block implements what we call TV-TV minimization, which enforces measurement consistency while guaranteeing that assumptions 1) and 2) are met.

1) TV-TV Minimization: Given the LR image b and a HR image w, TV-TV minimization consists of

$$\begin{aligned} &\underset{x}{\text{minimize}} \ \|x\|_{\text{TV}} + \beta\|x - w\|_{\text{TV}} \\ &\text{subject to } Ax = b, \end{aligned}$$

where x Rn is the optimization variable, and 0 tradeoffs between assumptions 1) and 2). Indeed, the first

term in the objective of (3) encodes assumption 1), the second term assumption 2), and the constraints enforce measurement consistency. Robustness of our method is achieved by the framework itself. CNNs, being purely data-driven, suffer from generalization errors. A generalization error typically implies lack of consistency with the input LR image. Our method overcomes this by guaranteeing measurement consistency, and thus robustness with respect to generalization errors, via the constraints of TV-TV minimization. Of course, our assumptions 1)-2) can be easily modified to better capture the class of images to be super-resolved. We found that using TV semi-norms in the objective yielded better results. In addition, as these functions are convex, problem (3) is convex as well. Although a problem like (3) has appeared before in [47] in the context of dynamic computed tomography (CT), the prior information w there was an image reconstructed by solving the same

problem in the previous instant; see also [48]–[50]. Our approach

is conceptually different in that we use (3) to improve the output of a CNN-based method. Next, we show how TV-TV minimization relates to 1-1 minimization, and how the theory for the latter in [48] suggeststhat selecting = 1 in (3) may lead to better performance.

4.3 Algorithm for TV-TV Minimization

We now explain how to efficiently solve TV-TV minimization (3) with ADMM [23]. In contrast with the majority of SR algorithms, which operate on individual patches, our algorithm operates on full images. We do that by capitalizing on the fact that matrix-vector multiplications can be performed fast whenever the matrix is D [cf. (2)] or any of the instantiations of A mentioned in Section III-A.

ADMM: The problem that ADMM solves is

$$\begin{aligned} &\underset{x}{\text{minimize}} \ \|x\|_{\text{TV}} + \beta\|x - w\|_{\text{TV}} \\ &\text{subject to } Ax = b, \end{aligned}$$

where f and g are closed, proper, and convex functions, and F and G are given matrices. Associating a dual variable to the constraints of (6), ADMM iterates on k

$$\begin{aligned} y^{k+1} &= \underset{y}{\text{argmin}} \ f(y) + \frac{\rho}{2}\|Fy + Gz^k + \lambda^k\|_2^2 \\ z^{k+1} &= \underset{z}{\text{argmin}} \ g(z) + \frac{\rho}{2}\|Fy^{k+1} + Gz + \lambda^k\|_2^2 \\ \lambda^{k+1} &= \lambda^k + Fy^{k+1} + Gz^{k+1}, \end{aligned}$$

where $\rho > 0$ is the augmented Lagrangian parameter.

4.4 Code

```
% ------------------------------------------------------------------------
% Simple experiment, using only two images and SRCNN
% ------------------------------------------------------------------------
% clear
main('../images/SampleImages/SampleImages_SRCNN', ...
  '../images/SampleImages/SampleImages_HR', 2, 'Y', 'ResultsTest.mat')

% ------------------------------------------------------------------------
------------
```

```
% Create a folder 'Test_Sets_Side' and download the images from the Google drive
link   provided
% Uncomment the following to replicate all the experiments in the paper
% ------------------------------------------------------------------------------
------------

% main('../images/Test_Sets_Side/SRCNNSet5x2/', ...
% '../images/Test_Sets_GT/Set5/Set5x2', 2, 'Y', 'SRCNNSet5x2_Results.mat');

function [] = main(path_side_info, path_groundtruth, ...
      scaling_factor, in_type, filename_results,  ...
      varargin)


% Loops through the images in the folders 'path_side_info' and
% 'path_groundtruth', which should exist in equal numbers and
% have the same order in the corresponding folders, and applies
% the post-processing step proposed in
% Note: To ensure that the same image in the respective folders correspond
% to eachother, all the image names consist of the image number followed by the
% type of image (e.g., GT) which are then accessed in ascending order.
%
% That post-processing step consists of solving a TV-TV minimization
% problem
%
%    minimize    TV(x) + β TV(x - w)
%       x
%    subject to  A*x = b,
%% where β is a trade-off parameter which we set to 1, w is the output
% of a given super-resolution algorithm, e.g., a CNN, assumed stored
% as an image file in the folder 'path_side_info', and A is a
% downsampling operator (default: Matlab's imresize) which is used
% to generate a low-resolution image b from the ground truth images
% asumed stored as an image file in the folder 'path_groundtruth'.
%% Inputs:
%   - 'path_side_info': string containing the path to a directory with images
%     corresponding to w, i.e., the output of a super-resolution algorithm.
%     To use the available outputs for example SRCNN with a scaling factor
```

```
%     of 4 for Set5: First create a folder 'Test_Sets_Side' and download the
%     corresponding files from the Google Drive links provided and set the
%     path to images/Test_Sets_Side/SRCNNNSet5x4. If considering the outputs

%     from DRCN for a scaling factor of 2 for BSD100, the path is
%     images/Test_Sets_Side/DRCNBSD100x2 etc.
%
%   - 'path_groundtruth': string containing the path to a directory
%     with the ground truth (high-resolution) images which are used to
%     obtain the low-resolution image b. For instance, to use the
%     available ground truth images of Set5 for a scaling factor of 2,
%     the path should read images/Test_Sets_GT/Set5_x2_GT. Note that,
%     the ground truth images are the same for each algorithm, it just
%     differs for different scaling factors.
%
%   - 'scaling_factor': whole numbers between 1 and 8 corresponding
%     to the upscaling factor. In our paper, we only consider scaling
%     factors x2, x4 and when possible x8 thus we only provide the
%     outputs for these factors. To use other scaling factors, store
%     the respective images. Note that, for example, a scaling_factor
%     of 2 means 4 times more pixels, as both the width and height are
%     augmented by 2.
%
%   - 'filename_results': string with the name of the file where the
%     results (PSNR and SSIM) will be stored.
%
% Optional inputs: pair 'string', value, where 'string' can be
%   - 'GPU': 0 or 1. If 1, the function uses code optimized for GPUs
%     (default: 0)
%
%   - 'SHOW_IMAGES': 0 or 1: If 1, processed images are displayed.
%     (default: 0)
%
%   - 'SHOW_RESULTS': 0 or 1: If 1, display PSNR and SSIM values.
```

```
%     (default: 0)
%
%   - 'BETA': the β value for TV-TV minimization where β > 0.

%     (default: 1)
%   - 'A_FORWARD': function handle that implements A*x, where x is a
%     vectorized version of a high-resolution image.
%     (default: the function in utils/A_bicubic.m, which uses Matlab's
%     imresize and uses the bicubic kernel. To use box filter  define the
%     this  optional input to the main function -  'A_FORWARD', 'A_box.m'
%     when calling it from the script experiment.m)
%
%   - 'A_TRANSPOSE: function handle that implements A'*y, where y is
%     a vectorized version of a low-resolution image.
%     (default: the function in utils/AT_bicubic.m, which uses Matlab's
%     imresize and uses the bicubic kernel. To use box filter  define the
%     this  optional input to the main function 'A_TRANSPOSE', 'AT_box.m'
%     when calling it from the script experiment.m)
%
% Path to utility functions
addpath('utils/')

% ================================================
% Check inputs

% ------------------------------------------------
% Check that 'path_side_info' and 'path_groundtruth' have the same
% number of images
ext   =  {'*/.jpg','/*.png','/*.bmp'}; % possible image extensions

files_path_side_info = [...
   dir([path_side_info, ext{1}]), ...
   dir([path_side_info, ext{2}]),  ...
   dir([path_side_info, ext{3}])
```

```
    ];
files_path_groundtruth = [...
    dir([path_groundtruth, ext{1}]), ...

 dir([path_groundtruth, ext{2}]),  ...
    dir([path_groundtruth, ext{3}])
    ];

if length(files_path_side_info) ~= length(files_path_groundtruth)
    error('Specified folders should have the same number of images')
end

if scaling_factor <= 1 || scaling_factor >= 9
    error('scaling_factor has to be between 1 and 8')
end

% ***********
% Defaults
% ***********
GPU = 0;
SHOW_IMAGES = 1;
SHOW_RESULTS = 1;
A_h  = @A_bicubic;
AT_h = @AT_bicubic;
beta = 1;

% Read optional input
if (rem(length(varargin),2) == 1)
    error('Optional parameters should always go in pairs');
else
    for i=1:2:(length(varargin)-1)
        switch upper(varargin{i})
            case 'GPU'
                GPU = varargin{i+1};
```

```
            case 'SHOW_IMAGES'
                SHOW_IMAGES = varargin{i+1};
            case 'SHOW_RESULTS'
                SHOW_RESULTS = varargin{i+1};
             case 'BETA'
                beta  = varargin{i+1};
            case 'A_FORWARD'
                A_h = varargin{i+1};
            case 'A_TRANSPOSE'
                AT_h = varargin{i+1};
            otherwise
                error(['Unrecognized option: ''' varargin{i} '''']);
        end
    end
end
% ====================================================

% ====================================================
% Main loop over all images

try
    % Images to be used as side information (w)
    DNNs = ReadImages(path_side_info);

    % Ground truth images (x)
    HR = ReadImages(path_groundtruth);
catch
    error('Something wrong with the folders used as input to main.m')
end

num_images = numel(HR);   % Number of images in the folders

% check if the images have the same size in the correspondig folders
for n = 1:1:num_images
```

```
   check(n) =  isequal(size(DNNs(n).data),size(HR(n).data));
end

if all(check) == 0
    fprintf('The images in the different folders do not correspond to eachother.')
end

psnr_tvtv = zeros(1, num_images);
ssim_tvtv = zeros(1, num_images);
psnr_cnn  = zeros(1, num_images);
ssim_cnn  = zeros(1, num_images);

 disp(['-----Super-Resolution-----', 'x', num2str(scaling_factor),'-----']);


parfor j = 1:1:num_images
    % Image w
    im_gt = struct()
    im_w = struct()
    im_HR = struct()
    im_w.out = DNNs(j).data;

    % if the original network is trained on an RGB image, no pre-processing
    % is done


    if in_type == 'RGB'
        w = im_w.out;
        x = HR(j).data;
      if size(im_w.out,3) > 1
        im_gt.out = im2double(HR(j).data(:,:,1));
        x =  im2double(reshape(x,[],3));
        w =  im2double(reshape(w, [], 3));
      end
```

```
 if size(im_w.out,3) == 1
    w = im_w.out;
    x = HR(j).data;
    im_gt.out = (HR(j).data);
    x =  im2double(x(:));
    w =  im2double(w(:));
 end
end

 if in_type == 'Y'
  if  size(im_w.out,3) > 1
   im1 = rgb2ycbcr(im_w.out);
   im_w.out = im1(:, :, 1);
   im_HR.out = HR(j).data;
   im1 = rgb2ycbcr(im_HR.out);
   im_gt.out = im1(:, :, 1);

  end

if size(HR(j).data,3) == 1
   im_gt.out = HR(j).data;
end
   w = im2double(im_w.out(:)); % normalizing the intensity values
   x = im2double(im_gt.out(:)); % normalizing the intensity values
end

% Dimensions of image
[M, N, channel] = size(im_w.out);
if in_type == 'RGB' & size(im_w.out,3) > 1
   channel = 3
else
  channel = 1
end
```

```
n = M*N;

% ------------------------------------------------------------

% ---------------------------------------------

%% Post-processing step using TVTV Solver
 X_ADMM = zeros(n,channel);
tic
for i = 1:channel
    % Obtain the LR image b by sampling x
   b = A_h(x(:,i),scaling_factor,M,N);
   if GPU
      [x_ADMM, k_ADMM] = TVTV_Solver_GPU(M, N, b, w(:,i), beta, A_h, AT_h,
scaling_factor);
      X_ADMM(:,i) = x_ADMM;
   else
      [x_ADMM, k_ADMM] = TVTV_Solver_CPU(M, N, b, w(:,i), beta, A_h, AT_h,
scaling_factor);
      X_ADMM(:,i) = x_ADMM;
   end

end
toc
fprintf('Image %i processed \n',j)
% ------------------------------------------------------------------------------------------------


% ------------------------------------------------------------------------------------------------
%% Reshape from vector to matrix and rescale entries to [1,255]
x_hat = (reshape(X_ADMM,M,N, channel));
x_hat = uint8(x_hat*255);

% ------------------------------------------------------------------------------------------------
```

```
%     if SHOW_IMAGES
%         plotimages(j,im_HR.out, im_gt.out,im_w.out, x_hat, scaling_factor)
%     end

% plotimages(j,im_HR.out, im_gt.out,im_w.out, x_hat, scaling_factor)

   if in_type == 'RGB' & size(im_w.out,3) > 1
      x_hat = rgb2ycbcr(x_hat);
      x_hat = x_hat(:,:,1);
      im1 = rgb2ycbcr(HR(j).data);
      im_gt.out =im1(:, :, 1);
   end

   %% Compute the PSNR and SSIM values
   [psnr_tvtv(j), ssim_tvtv(j)] = compute_diff(x_hat, im_gt.out, scaling_factor);
   [psnr_cnn(j), ssim_cnn(j)]   = compute_diff(im_w.out, im_gt.out, scaling_factor);

   % Print results
   if SHOW_RESULTS
      fprintf('Image %2d: TVTV - PSNR: %2.4f dB - SSIM: %2.4f \n', j, psnr_tvtv(j), ssim_tvtv(j))
      fprintf('        : CNN  - PSNR: %2.4f dB - SSIM: %2.4f \n', psnr_cnn(j) , ssim_cnn(j))
   end

end

% =====================================================
% Compute the mean over all images of PSNR and SSIM
psnr_tvtv_mean = mean(psnr_tvtv)
psnr_cnn_mean  = mean(psnr_cnn)
ssim_tvtv_mean = mean(ssim_tvtv)
ssim_cnn_mean  = mean(ssim_cnn)

%% Storing the results for each image and saving the results in a matrix
```

```
save(filename_results, 'psnr_tvtv', 'psnr_cnn', 'ssim_tvtv', 'ssim_cnn',  ...
'psnr_tvtv_mean', 'psnr_cnn_mean', 'ssim_tvtv_mean', 'ssim_cnn_mean')

% plotimages(j,im_HR.out, im_gt.out,im_w.out, x_hat, scaling_factor)
% ====================
```

RESULTS ANALYSIS

EXPERIMENTS

We now describe our experiments. After explaining the experimental setup, we expand on the phenomenon described in Fig. 1. Then, we consider the case of operator mismatches (i.e., A is different during training and testing), and show how our framework adds significant robustness in this scenario. Finally, we report experiments on standard SR datasets.

5.1 Experimental Setup

1) **Algorithm Parameters**: Most experiments were run using the same algorithm settings, unless indicated otherwise. The hyperparameter β in (3) was generally set to 1 and in some instances set to 2. For most experiments, A was the bicubic operator via MATLAB's IMRESIZE. For ADMM, we adopted the stopping criterion in [23, §3.3.1] with $\epsilon^{\text{pri}} = \epsilon^{\text{dual}} = 0.001$, or stopped after 500 iterations. Also, we initialized $\rho = 0.5$ and adjusted it automatically using the heuristic in [23, §3.4.1].

2) ***Datasets:*** We considered the standard SR test sets Set5 [52], Set14 [53], BSD100 [54] and Urban100 [55], which contain images of animals, buildings, people, and landscapes.

*3)****Computational Platform:*** All experiments were run on Matlab using a workstation with 12 core 2.10GHz Intel Xeon Silver 4110 CPU and two NVIDIA GeForce RTX GPUs.

4) ***Methods Evaluated****:* We compared our framework against the state-of-the-art methods in Table I and also considered simple TV minimization, i.e., (3) with β 0, using the TVAL3 solver [8]. The table shows the acronyms and references of the methods, their main technique, the scaling factors considered in the original papers, and the datasets used for training. Both during training and testing, SRCNN, DRCN and FSRCNN extract the luminance channel of the YCbCr colorspace, while the rest of the CNN-based methods in Table I work on all the RGB channels. During training, the HR images are converted to LR images by applying MATLAB's IMRESIZE, as originally done in [15]. The output images for SRCNN [15] were retrieved from an online repository.3 For the remaining methods, we generated the outputs from the available pretrained models.

5) **Performance Metrics**: We compared different algorithms by evaluating the PSNR (dB) and SSIM [61] on the luminance channel of the output images. We also provide sample images for qualitative evaluation.

TABLE I

METHODS USED IN OUR EXPERIMENTS. FOR EACH, WE SHOW THE MAIN TECHNIQUE, THE SCALING FACTORS IT CAN HANDLE AND, IF ANY, THE TRAINING DATASET

Method	Type	S.F.	Training dataset
SRCNN [15]	CNN	2, 4	ImageNet [18]
FSRCNN [26]	CNN	2, 4	T91 [13], General100 [26]
DRCN [35]	CNN	2, 4	T91 [13]
VDSR [17]	CNN	2, 4	T91 [13], BSDS200 [56]
LapSRN [27]	CNN	2, 4	T91 [13], BSDS200 [56]
SRMD [21]	CNN	2, 4	DIV2K [57], BSDS200 [56] WED [58]
RCAN [25]	CNN	2, 4	DIV2K
EDSR [24]	CNN	2, 4	DIV2K
IRCNN [28]	Plug-and-play	2, 4	ImageNet [18], WED [58], BSD500 [56]
$\text{ESRGAN}_{\text{PSNR}}$ [16]	GAN	4	DIV2K [57], Flickr2K [57], OutdoorSceneTraining [60]
DeepRED [42]	Plug-and-play	2, 4	
TVAL3 [8]	Optimization	2, 4	

5.2 Measurement Inconsistency of CNNs

We show that the phenomenon illustrated in Fig. 1 for SRCNN [15] occurs not only for this network, but is pervasive. That is, CNNs for SR fail to enforce measurement consistency (1) during testing. We chose three images for this purpose: Baboon from Set14, 38092 from BSD100, and img005 from Urban100. Every image is downsampled with MATLAB's IMRESIZE, which is the procedure executed for training each CNN, and the resulting LR image is fed into the network. We chose a scaling factor of 4.

Results: Table II shows the results for a subset of methods in Table I. In the 3rd column, it displays the 2-norm of the difference between the downsampled HR outputs, i.e., Aw, and the input LR image b; in the 4th column, it shows the same quantity after feeding the corresponding w (and b, cf. Fig. 2) to our method. It can be seen that our post-processing improves consistency by 6 orders of magnitude. Note that even though SRMD models various degradations without retraining, it still fails to ensure consistency.

TABLE II CONSISTENCY ACHIEVED BY CNN-TYPEMETHODS $(\|Aw-b\|_2)$ AND BY OUR ALGORITHM $(\|A\hat{x}-b\|_2)$

Method	Image	$\|Aw - b\|_2$	$\|A\hat{x} - b\|_2$
SRCNN [15]	*Baboon*	5.29×10^{-1}	$\mathbf{4.77 \times 10^{-7}}$
	38092	4.32×10^{-1}	$\mathbf{3.54 \times 10^{-7}}$
	img_{005}	14.93×10^{-1}	$\mathbf{8.02 \times 10^{-7}}$
FSRCNN [26]	*Baboon*	3.26×10^{-1}	$\mathbf{4.93 \times 10^{-7}}$
	38092	2.91×10^{-1}	$\mathbf{4.07 \times 10^{-7}}$
	img_{005}	10.32×10^{-1}	$\mathbf{4.10 \times 10^{-7}}$
SRMD [21]	*Baboon*	4.14×10^{-1}	$\mathbf{6.85 \times 10^{-7}}$
	38092	2.39×10^{-1}	$\mathbf{9.96 \times 10^{-7}}$
	img_{005}	9.53×10^{-1}	$\mathbf{3.95 \times 10^{-7}}$
IRCNN [28]	*Baboon*	1.09×10^{-1}	$\mathbf{6.14 \times 10^{-7}}$
	38092	9.15×10^{-2}	$\mathbf{5.12 \times 10^{-7}}$
	img_{005}	5.33×10^{-1}	$\mathbf{7.15 \times 10^{-7}}$
DeepRed [43]	*Baboon*	2.62×10^{-1}	
	38092	2.06×10^{-1}	
	img_{005}	3.11×10^{-1}	

5.3 Robustness to Operator Mismatch As previously stated, most SR CNNs are trained by downsampling a HR into a LR image using the bicubic operator. If, during testing, A is different from the bicubic operator then, as we will see, there can be a serious drop in performance. This may indeed limit the applicability of CNNs in real-life scenarios where the required time and computation resources might not be available. Our approach, however, mitigates this effect and adds robustness to the SR task. We considered the same images and methods as in Table II, with DeepRed replaced by TVAL3, and considered the operators for A described in Section III-A: bicubic, box averaging, and simple subsampling.

Results: Each shaded box in Table III shows, for each subsampling operator, the PSNR values obtained by a given method, and by subsequently processing its output with our scheme. While all methods perform the best under bicubic subsampling, there is a performance drop for box filtering, and an even larger drop for simple subsampling. Note that our method systematically improves the output of all the networks, even for bicubic subsampling. And while the improvement is of less than 1dB for bicubic subsampling, it averages around 2dBs for simple subsampling. Indeed, the performance of the CNNs for this case drops so much that there is a large margin for improvement. Interestingly, TVAL3, which solves (3) with $\beta = 0$, is the worst method for bicubic subsampling, but mapproaches the performance of CNNs for box averaging and, besides ours, becomes the best for simple subsampling. Hence, this illustrates that reconstruction-based methods can be more robust and adaptable than CNN architectures. We also conducted more systematic experiments using the standard datasets Set5, Set14, BSD100, and Urban100, under different scaling factors and using bicubic downsampling only.

TABLE III OPERATOR MISMATCH EXPERIMENTS. PSNR VALUES UNDER DIFFERENT SAMPLING OPERATORS

Method	Image	Bic.	*Ours*	Box	*Ours*	Sub.	*Ours*
SRCNN [15]	*Baboon*	22.70	**22.73**	22.49	**22.53**	17.48	**19.16**
	38092	25.90	**25.95**	25.69	**25.75**	20.07	**21.84**
	img_{005}	25.12	**25.27**	24.99	**25.21**	17.92	**19.70**
FSRCNN [26]	*Baboon*	22.79	**22.80**	22.49	**22.55**	17.38	**19.28**
	38092	26.03	**26.05**	25.64	**25.73**	20.00	**21.94**
	img_{005}	25.81	**25.85**	25.12	**25.34**	17.79	**19.72**
SRMD [21]	*Baboon*	22.90	**22.91**	22.52	**22.59**	16.95	**18.98**
	38092	26.20	**26.21**	25.62	**25.73**	19.39	**21.50**
	img_{005}	26.56	**26.61**	25.59	**25.89**	17.36	**19.30**
IRCNN [28]	*Baboon*	**22.76**	**22.76**	**22.51**	**22.51**	17.41	**19.45**
	38092	**26.09**	**26.09**	25.77	**25.78**	20.54	**22.40**
	img_{005}	26.18	**26.21**	**25.86**	**25.86**	18.23	**19.64**
TVAL3 [8]	*Baboon*	22.40		22.27		20.83	
	38092	25.59		25.00		21.35	
	img_{005}	24.29		22.54		17.28	

5.4 Quantitative Results: Table V displays the average PSNR and SSIM, as well as the average execution time of our method (in seconds), for 2×, and 4× scaling factors. Each shaded area shows the performance of a given (learning-based) reference method, the performance of our scheme applied to the output of that reference method, and the average execution time (of our method).

After executing in the MATLAB,

-----Super-Resolution-----x2-----

Starting parallel pool (parpool) using the 'local' profile ... Connected to the parallel pool (number of workers):

Elapsed time is 229.774559 seconds.

Image 2 processed

Image 2: TVTV - PSNR: 32.4788 dB - SSIM: 0.9642

: CNN - PSNR: 32.1808 dB - SSIM: 0.9596

Elapsed time is 283.622641 seconds.

Image 1 processed

Image 1: TVTV - PSNR: 40.8670 dB - SSIM: 0.9854

: CNN - PSNR: 40.5123 dB - SSIM: 0.9838

psnr_tvtv_mean = 36.6729

psnr_cnn_mean = 36.3466

ssim_tvtv_mean = 0.9748

ssim_cnn_mean = 0.9717

TABLE V AVERAGE PSNR (SSIM) RESULTS IN dB AND EXECUTION TIME IN SECONDS OF OUR METHOD USING THE REFERENCE METHODS

Dataset	Scale	TVAL3 [8]	SRCNN [15]	*Ours*	Time	FSRCNN [26]	*Ours*	Time
Set5	×2	34.0315 (0.9354)	36.2772 (0.9509)	**36.5126** (**0.9535**)	18.23	36.9912 (0.9556)	**37.0368** (**0.9559**)	8.97
	×4	29.1708 (0.8349)	30.0765 (0.8525)	**30.2460** (**0.8583**)	13.93	30.7122 (0.8658)	**30.7886** (**0.8686**)	9.86
Set14	×2	31.0033 (0.8871)	32.1245 (0.9028)	**32.2793** (**0.9055**)	11.57	32.6516 (0.9089)	**32.6880** (**0.9091**)	8.92
	×4	26.6742 (0.7278)	27.1808 (0.7410)	**27.2952** (**0.7476**)	8.21	27.6179 (0.7550)	**27.6794** (**0.7569**)	9.04
BSD100	×2	30.1373 (0.8671)	31.1087 (0.8835)	**31.2148** (**0.8864**)	6.79	31.5075 (0.8905)	**31.5229** (**0.8907**)	4.21
	×4	26.3402 (0.6900)	26.7027 (0.7018)	**26.7793** (**0.7082**)	5.30	26.9675 (0.7130)	**26.9966** (**0.7146**)	4.64
Urban100	×2	27.5143 (0.8728)	28.6505 (0.8909)	**28.8219** (**0.8935**)	25.34	29.8734 (0.9010)	**29.8926** (**0.9013**)	27.75
	×4	23.7529 (0.6977)	24.1443 (0.7047)	**24.2308** (**0.7110**)	24.12	24.6196 (0.7270)	**24.6522** (**0.7291**)	25.08

Dataset	Scale	TVAL3 [8]	DRCN [35]	*Ours*	Time	VDSR [17]	*Ours*	Time
Set5	×2	34.0315 (0.9354)	37.6279 (0.9588)	**37.6697** (**0.9591**)	20.40	37.5295 (**0.9587**)	**37.5397** (0.9585)	65.91
	×4	29.1708 (0.8349)	31.5344 (0.8854)	**31.5660** (**0.8857**)	14.25	31.3485 (**0.8838**)	**31.3696** (0.8836)	42.38
Set14	×2	31.0033 (0.8871)	33.0585 (0.9121)	**33.1033** (**0.9129**)	8.92	33.0527 (0.9127)	**33.0906** (**0.9128**)	20.05
	×4	26.6742 (0.7278)	28.0269 (0.7673)	**28.0551** (**0.7679**)	8.57	28.0152 (0.7678)	**28.0375** (**0.7679**)	19.25
BSD100	×2	30.1373 (0.8671)	31.8536 (0.8942)	**31.8722** (**0.8952**)	26.23	31.9078 (**0.8960**)	**31.9071** (**0.8960**)	106.92
	×4	26.3402 (0.6900)	27.2364 (0.7233)	**27.2491** (**0.7239**)	24.23	26.8776 (0.7093)	**27.2342** (**0.7228**)	109.86

Dataset	Scale	TVAL3 [8]	LapSRN [27]	*Ours*	Time	SRMD [21]	*Ours*	Time
Set5	×2	34.0315 (0.9354)	37.7008 (0.9590)	**37.7132** (**0.9592**)	52.80	37.4496 (0.9579)	**37.5859** (**0.9588**)	51.96
	×4	29.1708 (0.8349)	31.7181 (0.8891)	**31.7428** (**0.8894**)	49.45	31.5750 (0.8853)	**31.6695** (**0.8869**)	37.60
	×8	21.7360 (0.6360)	26.3314 (**0.7548**)	**26.3876** (0.7544)	42.50			
Set14	×2	31.0033 (0.8871)	33.2518 (0.9138)	**33.2618** (**0.9141**)	53.57	32.9460 (0.9126)	**33.2139** (**0.9140**)	51.43
	×4	26.6742 (0.7278)	28.2533 (0.7730)	**28.2682** (**0.7731**)	36.43	28.0833 (0.7721)	**28.2229** (**0.7725**)	36.42
	×8	20.8616 (0.5676)	24.5643 (**0.6266**)	**24.5946** (**0.6266**)	24.54			
BSD100	×2	30.1373 (0.8671)	32.0214 (0.8970)	**32.0281** (**0.8975**)	19.56	31.8722 (0.8953)	**31.9032** (**0.8960**)	23.71
	×4	26.3402 (0.6900)	27.4164 (0.7296)	**27.4298** (**0.7298**)	18.12	27.3350 (0.7273)	**27.3608** (**0.7283**)	20.94
	×8	20.1400 (0.5575)	24.6495 (**0.5887**)	**24.6776** (**0.5887**)	19.32			
Urban100	×2	27.9935 (0.8742)	31.1319 (0.9180)	**31.1464** (**0.9183**)	115.23	30.8799 (0.9146)	**30.9314** (**0.9153**)	108.32
	×4	23.7529 (0.6977)	25.5026 (**0.7661**)	**25.5152** (**0.7661**)	104.23	25.3494 (0.7605)	**25.3889** (**0.7614**)	102.15
	×8	18.8295 (0.5436)	22.0547 (**0.5956**)	**22.0788** (0.5950)	103.29			

Dataset	Scale	TVAL3 [8]	EDSR [24]	*Ours*	Time	RCAN [25]	*Ours*	Time
Set5	×2	34.0315 (0.9354)	37.9022 (0.9594)	**37.9198** (**0.9597**)	61.84	38.1819 (0.9604)	**38.2121** (**0.9608**)	58.22
	×4	29.1708 (0.8349)	32.0726 (0.8927)	**32.0968** (**0.8931**)	43.10	32.6003 (0.8991)	**32.6137** (**0.8992**)	38.51
	×8	21.7360 (0.6360)				27.2985 (**0.7866**)	**27.3264** (0.7864)	41.36
Set14	×2	31.0033 (0.8871)	33.4433 (0.9162)	**33.4862** (**0.9167**)	52.29	33.9896 (0.9203)	**34.0168** (**0.9207**)	32.15
	×4	26.6742 (0.7278)	28.4719 (0.7790)	**28.4949** (**0.7797**)	36.43	28.7596 (0.7866)	**28.7898** (**0.7869**)	18.64
	×8	20.8616 (0.5676)				25.1276 (**0.6479**)	**25.1697** (0.6474)	22.31
BSD100	×2	27.9935 (0.8742)	32.1323 (0.8986)	**32.1423** (**0.8989**)	21.34	32.3825(0.9019)	**32.3912** (**0.9022**)	21.67
	×4	26.3402 (0.6900)	27.5479 (0.7349)	**27.5579** (**0.7354**)	16.02	27.7547 (0.7428)	**27.7766**(**0.7429**)	19.65
	×8	20.1400 (0.5575)				24.9745 (**0.6050**)	**24.9982** (0.6047)	18.58
Urban100	×2	27.9935 (0.8742)	32.6128 (0.9152)	**32.6248** (**0.9153**)	105.23	33.0248 (0.9327)	**33.0488** (**0.9328**)	106.54
	×4	26.3402 (0.6900)	26.0311 (0.7841)	**26.0434** (**0.7842**)	102.94	26.8132 (0.8075)	**26.8251** (**0.8080**)	107.34
	×8	18.8295 (0.5436)				23.0010 (**0.6445**)	**23.0183** (0.6438)	108.26

Dataset	Scale	TVAL3 [8]	IRCNN [28]	*Ours*	Time	$\text{ESRGAN}_{\text{PSNR}}$ [16]	*Ours*	Time
Set5	×2	34.0315 (0.9354)	37.3436 (0.9572)	**37.3712** (**0.9576**)	52.40			
	×4	29.1708 (0.8349)	30.9995 (0.8778)	**31.0082** (**0.8780**)	37.20	32.7072 (0.9001)	**32.7227** (**0.9002**)	37.33
Set14	×2	31.0033 (0.8871)	32.8573 (0.9105)	**32.8947** (**0.9109**)	51.43			
	×4	26.6742 (0.7278)	27.7195 (0.7614)	**27.7454** (**0.7617**)	37.50	28.8920 (0.7893)	**28.9151** (**0.7895**)	36.86
BSD100	×2	31.0033 (0.8871)	31.6543 (0.8918)	**31.6745** (**0.8923**)	25.04			
	×4	26.3402 (0.6900)	27.0848 (0.7188)	**27.0920** (**0.7191**)	20.79	27.8332 (0.7447)	**27.8531** (**0.7452**)	22.14
Urban100	×2	31.0033 (0.8871)	30.0623 (0.9105)	**30.0899** (**0.9108**)	101.32			
	×4	23.7529 (0.6977)	24.8913 (0.7395)	**24.9041** (**0.7396**)	101.27	27.0270 (**0.8146**)	**27.0308** (0.8142)	108.86

5.5 Qualitative Results: Figures 4 and 5 depict the output images of all the algorithms (except IRCNN) for the test images baboon from Set14, and img067 from Urban100. All super-resolved images exhibit blur and loss of information compared with the GT images in Figs. 4a-5a. And as our scheme builds upon the outputs of other methods, it also inherits some of their artifacts. It is difficult to visually assess differences between the outputs of the algorithms and of our method, in part because the improvements, as measured by the PSNR, are relatively small.

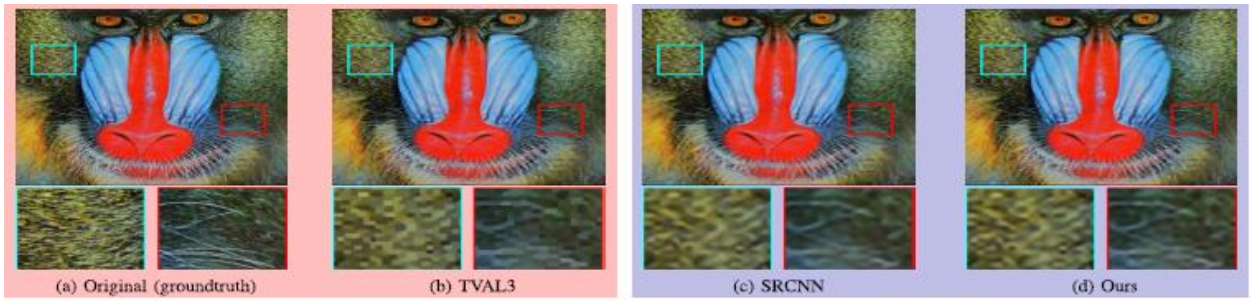


Fig. 4. Results on *Baboon* (Set14) for 4×. Each shaded area (except the top-left) shows the output of a learning-based algorithm and of our method.

Fig. 5. Results on *img076* (Urban100) for 4×. Each shaded area (except the top-left) shows the output of a learning-based algorithm and of our method.

CONCLUSION

We proposed a framework for single-image SR that blends model- and learning-based (e.g., CNN) techniques. As a result, our framework enables solving the consistency problem that CNNs suffer from, namely that downsampled output (HR) images fail to match the input (LR) images. Our experiments show that enforcing such consistency not only systematically improves the quality of the output images of CNNs, but also adds robustness to the SR task. At the core of our framework is a problem that we call TV-TV minimization and which we solve with an ADMM-based algorithm. Possible lines of future research include designing loss functions that enforce consistency during training and unrolling the proposed algorithm with a neural network.

FUTURE SCOPE

Deep Learning methods can be adopted to make low-quality content look better, sharpening the picture and removing noise and other artifacts caused by digital compression or analog conversion.

REFERENCES

1. M. Vella and J. F. C. Mota, "Single image super-resolution via CNN architectures and TV-TV minimization," in *Proc. BMVC*, 2019, pp. 1–12, doi: 10.5244/C.33.219.
2. M. Vella and J. F. C. Mota, "Robust super-resolution via deep learning and TV priors," in *Proc. SPARS*, 2019, pp. 1–3.
3. T. Blu, P. Thévenaz, and M. Unser, "Linear interpolation revitalized,"
4. *IEEE Trans. Image Process.*, vol. 13, no. 5, pp. 710–719, May 2004.
5. R. Fattal, "Image upsampling via imposed edge statistics," *ACM Trans. Graph.*, vol. 26, no. 3, p. 95, Jul. 2007.
6. C. Li, W. Win, H. Jing, and Y. Zhang, "An efficient augmented Lagrangian method with applications to total variation minimization," *Comput. Optim. Appl.*, vol. 56, no. 3, pp. 507–530, Dec. 2013.
7. S. Sun, D. Xue, and D. Chen, "Image magnification using fractional order level set reconstruction," in *Proc. 25th Chin. Control Decis. Conf. (CCDC)*, May 2013, pp. 333–334.
8. H. A. Aly and E. Dubois, "Image up-sampling using total-variation regularization with a new observation model," *IEEE Trans. Image Process.*, vol. 14, no. 10, pp. 1647–1659, Oct. 2005.
9. A. Chambolle, "An algorithm for total variation minimization and applications," *J. Math. Imag. Vis.*, vol. 20, no. 1, pp. 89–97, 2004.
10. S. Becker, J. Bobin, and E. J. Candès, "NESTA: A fast and accurate first-order method for sparse recovery," *SIAM J. Imag. Sci.*, vol. 4, no. 1,pp. 1–39, 2011.
11. J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Trans. Image Process.*, vol. 19, no. 11, pp. 2861–2873, Nov. 2010.
12. A. Singh, F. Porikli, and N. Ahuja, "Super-resolving noisy images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 2846–2853.
13. C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image

super-resolution," in *Proc. ECCV*, 2014, pp. 184–199.

14. X. Wang *et al.*, "ESRGAN: Enhanced super-resolution generative adver- sarial networks," in *Proc. ECCVW*, Sep. 2018, pp. 1–16.
15. J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. CVPR*, Jun. 2009, pp. 248–255.
16. M. Haris, G. Shakhnarovich, and N. Ukita, "Deep back-projection networks for super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1664–1673.
17. B. Ghojogh, F. Karray, and M. Crowley, "Backprojection for training feedforward neural networks in the input and feature spaces," in *Proc. ICIAR*, 2020, pp. 16–24.
18. K. Zhang, W. Zuo, and L. Zhang, "Learning a single convolutional super- resolution network for multiple degradations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3262–3271.
19. S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.
20. B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 136–144.
21. Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super- resolution using very deep residual channel attention networks," in *Proc. ECCV*, Sep. 2018, pp. 286–301.
22. C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *Proc. ECCV*, 2016, pp. 391–407.
23. W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep Laplacian pyramid networks for fast and accurate super-resolution," in *Proc. CVPR*, Jul. 2017, pp. 624–632.
24. K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep CNN denoiser prior for image restoration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3929–3938.
25. S. Mallat and G. Yu, "Super-resolution with sparse mixing estimators,"*IEEE Trans. Image Process.*, vol. 19, no. 11, pp. 2889–2900, Nov. 2010.
26. W. Zhang and W.-K. Cham, "Hallucinating face in the DCT domain," *IEEE Trans. Image*

Process., vol. 20, no. 10, pp. 2769–2779, Oct.